目　录

第一部分　快速入门——MATLAB语言精要

第二部分　精通 SIMULINK

第一部分
快速入门——MATLAB 语言精要

第一章 MATLAB 语言简介及其安装

1.1 MATLAB 语言简介

电子计算机的出现和发展是现代科学技术的巨大成就之一。它对科学技术的几乎一切领域，特别是对数值计算、数据处理、统计分析、人工智能以及自动控制等方面产生了极其深远的影响。熟练掌握利用计算机进行科学研究和工程应用的技术，已经成为广大科研设计人员必须具备的基本能力之一。

大部分从事科学研究和工程应用的读者朋友可能都已经注意到并为之所困扰的是，当我们的计算涉及矩阵运算或画图时，利用 FORTRAN 和 C 语言等计算机语言进行程序设计是一项很麻烦的工作。不仅需要对所利用的有关算法有深刻的了解，还需要熟练掌握所用语言的语法和编程技巧。例如对矩阵求逆这样的一个运算，我们首先要选择一个较好的求逆算法，然后利用 FORTRAN 或 C 语言等高级语言编程来逐步地实现此算法，且不说键入程序的枯燥和费事，等经过了艰苦烦琐的调试工作终于实现算法达到目的后，我们会发现，所编制的百余条甚至几百条语句仅仅是完成了一个矩阵的求逆工作，我们不免为自己的工作效率大发感叹。并不复杂的计算任务，用计算机来实现竟是如此的烦琐，面对手头要完成的研究任务，也许会产生畏惧之感。

MATLAB 正是为免除无数类似上述的尴尬局面而产生的。在 1980 年前后，美国的 Cleve Moler 博士在 New Mexico 大学讲授线性代数课程时，发现应用其它高级语言编程极为不便，便构思并开发了 MATLAB(MATrix LABoratory，即矩阵实验室)，它是集命令翻译、科学计算于一身的一套交互式软件系统，经过在该大学进行了几年的试用之后，于 1984 年推出了该软件的正式版本。在 MATLAB 下，矩阵的运算变得异常的容易，后来的版本中又增添了丰富多彩的图形图像处理及多媒体功能，使得 MATLAB 的应用范围越来越广泛，Moler 博士等一批数学家与软件专家组建了一个名为 MathWorks 的软件开发公司，专门扩展并改进 MATLAB，并于 1994 年推出了最新的 4.2 版本。

为了准确地把一个控制系统的复杂模型输入给计算机，然后对之进行进一步的分析与仿真，1990 年 MathWorks 软件公司为 MATLAB 提供了新的控制系统模型图形输入与仿真工具，并定名为 SIMULAB，该工具很快在控制界得到了广泛的使用。但因其名字与著名的软件 SIMULA 类似，所以在 1992 年正式改名为 SIMULINK。此软件有两个明显的功能：仿真与连接，亦即可以利用鼠标器在模型窗口上画出所需的控制系统模型，然后利用该软件提供的功能来对系统直接进行仿真。很明显，这种做法使得一个很复杂系统的输入变得相当容易。SIMULINK 的出现，更使得 MATLAB 为控制系统的仿真与其在 CAD 中的应用打开了崭新的局面。

目前的 MATLAB 已经成为国际上最为流行的软件之一，它除了传统的交互式编程之外，还提供了丰富可靠的矩阵运算、图形绘制、数据处理、图像处理、方便的 Windows 编程等便利工具，出现了各种以 MATLAB 为基础的实用工具箱，广泛地应用于自动控制、图像信号处理、

9810022

生物医学工程、语音处理、雷达工程、信号分析、振动理论、时序分析与建模、化学统计学、优化设计等领域，并表现出一般高级语言难以比拟的优势。较为常见的 MATLAB 工具箱主要包括：

(1) 控制系统工具箱(control systems toolbox)。

(2) 系统辨识工具箱(system identification toolbox)。

(3) 鲁棒控制工具箱(robust control toolbox)。

(4) 多变量频率设计工具箱(multivariable frequency design toolbox)。

(5) μ 分析与综合工具箱(μ - analysis and synthesis toolbox)。

(6) 神经网络工具箱(neural network toolbox)。

(7) 最优化工具箱(optimization toolbox)。

(8) 信号处理工具箱(signal processing toolbox)。

(9) 模糊推理系统工具箱(fuzzy inference system toolbox)。

(10) 小波分析工具箱(wavelet toolbox)。

1.2 MATLAB 软件的安装

早期的 MATLAB 版本是基于 DOS 操作系统的，随着软件技术的飞速发展，特别是新的操作系统 WINDOWS 的广泛流行，MATLAB 发展成可在 WINDOWS 下运行的版本。考虑到目前较为流行的版本是 MATLAB 4.2，本书中的所有叙述都是基于 MATLAB 4.x 的。

在 PC 兼容机下使用 MATLAB 的前提条件是该机器配备有协处理器芯片(math - coprocessor)，当然 486 DX 以上的机型因为协处理器在 CPU 上已经存在，所以不必另外配置该芯片。运行 MATLAB 4.x 版本的硬件条件更为苛刻，它要求首先运行 386 增强型的 Micro - soft Windows 环境，而且有 4 MB 以上的内存，要实现其中一些特殊的功能则需要 8 MB 以上的内存。

MATLAB 4.x 的基本部分有 5 张 3 吋软盘，欲进行安装，首先应该启动 Windows 环境，若机器中没有安装此环境，则需先进行 Windows 的安装。待以上的准备工作做好后，在计算机的软驱中插入 MATLAB 软件的第一片盘，然后利用 Windows 的文件管理器找到软盘上的 setup.exe，用鼠标双击该文件即可开始 MATLAB 的安装，根据安装程序的提示，依次将其余的几张盘插入软驱中，进行将各个盘上的内容复制到硬盘上的操作。

须要说明的是：

(1) 在安装的过程中，计算机会自动要求用户选择 MATLAB 的安装位置，即工作目录。缺省的位置是 C 盘下的 MATLAB 子目录，此目录并不需要事先存在，在安装时，安装程序会自动建立此目录。当然，用户也可将 MATLAB 安装到硬盘上的任何地方，只需在安装程序给出确定安装位置的提示时进行键入即可。

(2) 在路径确定好后，计算机将给出安装 MATLAB 所需的硬盘空间及准备安装位置的剩余空间大小这样两个信息框，用以帮助用户选择安装 MATLAB 的合适方案。这些可选的方案包括“全部安装”(Full MATLAB Installation)和“选择安装”(Custom MATLAB Installation)两种，若硬盘空间足够的话，我们推荐用户选择全部安装。

(3) 若用户选择了“选择安装”，则又分为以下的 4 种方案：

① Matlab Environment [6Meg]；② Image Demos [2Meg]；③ Ghost Script Support

[2Meg];④ Install additional Toolboxes。

第一种方案只给出最基本的 MATLAB 操作环境,方括号中指出其所需空间的字节数;第二种方案给出 MATLAB 的工作演示;第三种选择是支持图形的功能;第四种方案则是增加上次安装时没有装上的部分和扩充 MATLAB 的库函数。

待 MATLAB 4.x 基本部分的 5 张软盘都已安装完成之后,安装程序会给出是否立即进行 MATLAB 工具箱安装的提示,若想安装工具箱,可选择"Yes",依据提示插入工具箱软盘并回车,这种操作可一直进行下去直到你再也不想安装工具箱为止。当然也可选择"No"不进行工具箱的安装。可在以后的任意时刻添加工具箱,关于工具箱的添加后面章节会有说明,此处不再赘述。

以上的工作完成之后,返回到 Windows 的程序管理器窗口中,将会看到一个 MATLAB 程序组窗口已经在安装过程中自动建立。MATLAB 的启动程序 matlab.exe(描述为 Matlab)、有关 MATLAB 说明书的更正信息 README 和实时帮助功能 MATLAB HELP 三个选项存在于此程序组窗口中。用鼠标双击 Matlab 图标,就可进入 MATLAB ,出现 MATLAB 的命令窗口,如图 1-1 所示。

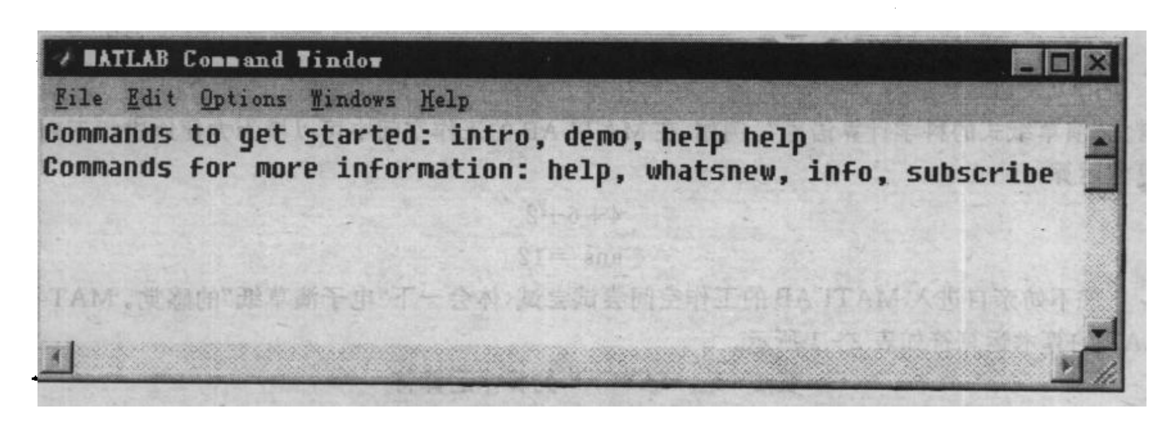


图 1-1 MATLAB 的命令窗口

要注意的一点是:安装完毕后,依据 Windows 是中文的或西文的不同,MATLAB 的命令提示符也有所不同,在西文环境下的提示符为>>,而在中文环境下则没有提示符。图 1-1 给出的是中文环境下的 MATLAB 命令窗口。

出现 MATLAB 的命令窗口后,即可在其上亲身感受 MATLAB 的便利之处了。

第二章　MATLAB 语言的基本使用方法

MATLAB 语言可以被认为是一种解释性语言，用户可以在 MATLAB 的工作空间中键入一个命令，也可以应用 MATLAB 语言编写应用程序，这样 MATLAB 软件对此命令或程序中的各条语句进行翻译，然后在 MATLAB 环境中对它进行处理，最后返回运算结果。

MATLAB 语言由早期专门用于矩阵运算的计算机语言发展而来，这正如其名称——"矩阵实验室"（Matrix Laboratory）的含义一样。它最基本、也是最重要的功能就是"进行实数矩阵或复数矩阵的运算"。因向量可作为矩阵的一列或一行，标量(一个数)有时则作为只含一个元素的矩阵，故向量和标量都可以作为特殊矩阵来处理。MATLAB 的操作和命令对于矩阵来说，并不完全等同于我们平时使用的形式，而是有着它自己的规定。

2.1　MATLAB 语言的基本特性

2.1.1　演草纸式的数学运算

用 MATLAB 进行数学运算，就像在计算器上算算术一样简单方便。因此，MATLAB 被誉为"演草纸式的科学计算语言"。例如，在 MATLAB 的工作空间中可以极为方便地进行下列算术运算：

```
4+6+2
ans =12
```

您不妨亲自进入 MATLAB 的工作空间尝试尝试，体会一下"电子演草纸"的感觉。MATLAB 的算术运算符如表 2-1 所示。

表 2-1　MATLAB 的算术运算符

算术运算	算术运算符	示　例
加法，a+b	+	5+3
减法，a−b	−	23−12
乘法，a×b	*	3.14 * 0.85
除法，a÷b	/ 或者 \	56/8 = 8\56
乘方，a^b	^	5 ^ 2

2.1.2　MATLAB 的工作空间

MATLAB 4.2 C 版是一个高度集成的语言环境，在其工作空间(Workspace)中可以编写程序、运行程序并跟踪调试程序。从图 2-1 可以看出 MATLAB 工作空间的界面下有一个菜单条，每个菜单选项又有子菜单，其中提供了很多有用的功能。

输入 who 命令可检查在工作空间中所建立的变量名。例如，如果当前工作空间中的变量为：ans、bananas、fruit、apples、cantaloupes，那么，在工作空间中键入 who，显示结果为：

```
who
```

Your variables are:

ans　　　bananas　　　fruit

apples　　　cantaloupes

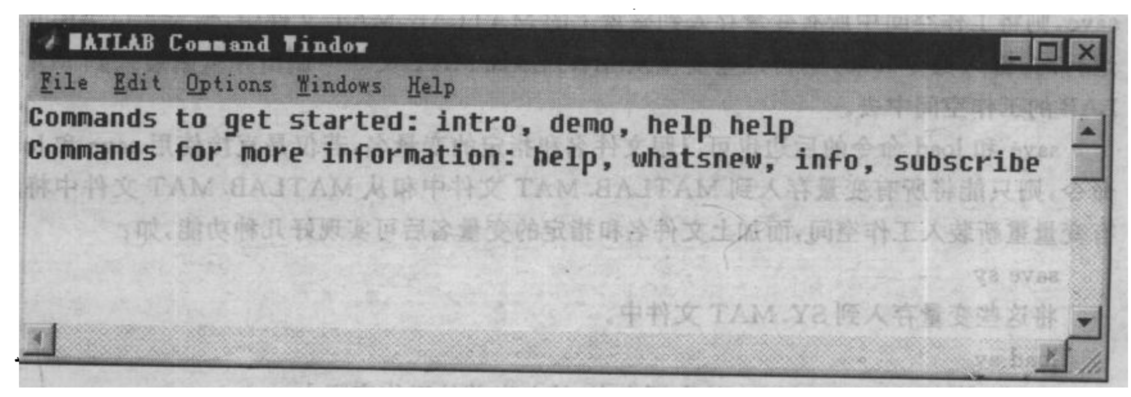


图 2-1　MATLAB 的工作空间

在 MATLAB 的变量空间中，还存在一些固定变量，如 eps、pi、Inf、NaN 等。变量 eps 在决定诸如奇异性和秩时，可作为一个容许误差，如 eps $=2^{-25}$，即精确到 2.22×10E－16，另外用户也可将此变量置为包括零值的其它任务值；变量 pi 即为 π，可由程序 4 * atan(1)计算得到，也可以用方法 imag(log(－1))得到。输入 imag(log(－1))，结果为：ans＝3.141 6。

Inf 表示正无穷大，产生 Inf 的方法：输入 1/0，结果为：

Warning:Divide by zero

ans＝

Inf

2.1.3　数据的存储和调用

在 MATLAB 的工作空间中可以方便地将数据存成文件，也可以随时调用数据文件。选择工作空间中 File 菜单项的子菜单项 Save Workspace as...，弹出一个标准的对话框，如图 2-2 所示。

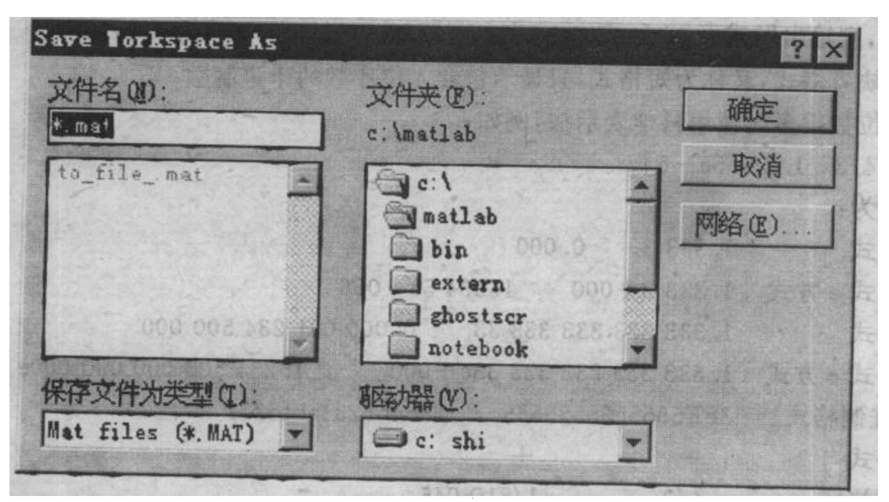


图 2-2　对话框

同样，选择工作空间中 File 菜单项的 Load Workspace... 子菜单项，可以方便地调用存储的数据文件。

数据的存储和调用，除了菜单操作之外，还可以直接在工作空间中输入命令。键入命令 save，则将工作空间中所有变量存入到磁盘上的 MATLAB.MAT 文件中，当 MATLAB 再被运行时，键入命令 load，则将这些变量从 MATLAB.MAT 文件中调出并重新装入到 MATLAB 的工作空间中去。

save 和 load 命令的后边也可以跟文件名和指定的变量名，若仅是直接使用 save 和 load 命令，则只能将所有变量存入到 MATLAB.MAT 文件中和从 MATLAB.MAT 文件中将所有变量重新装入工作空间，而加上文件名和指定的变量名后可实现好几种功能，如：

save sy

将这些变量存入到 SY.MAT 文件中。

load sy

将这些变量从 SY.MAT 文件中调出来，放入当前的工作空间中。

若欲存入指定的变量到某个文件中，可使用以下命令：

save sy x

仅存入变量 x 到 SY.MAT 文件中。

save sy x y z

存入变量 x，y 和 z。

lode sy x

将存入的指定变量 x 从 SY.MAT 文件中重新调出到当前工作空间中。load 也可读 ASCI 文件。

2.1.4 数据显示格式

任何 MATLAB 语句的执行结果都可在屏幕上显示，同时赋值给指定的变量，没有指定变量时赋值给一个特殊的变量 ans，数据显示格式可由 format 命令来控制。format 只影响结果的显示，不影响其计算与存储。MATLAB 总是以双精度执行所有运算。如果矩阵元素是整数，则矩阵显示没有小数。如： x=[−1 0 1]，显示结果为 x=−1 0 1 。如果矩阵元素不是整数，则输出形式有：

(1) 缺省格式，又称为短格式，只显示包含 4 位小数的十进制数。

(2) 位数很多时使用科学表示法，例如：

x=[4/3 1.234 5e−6]

输出结果为：

短格式	1.333 3	0.000
短格式 e 方式	1.333 3+000	1.234 5e−006
长格式	1.333 333 333 333 33	0.000 001 234 500 000
长格式 e 方式	1.333 333 333 333 33e+000	1.234 500 000 000 000e−006
16 进制格式	3FF5555555555555	3eb4b6231abfd271
十格式	+	+
分数格式	4/3	1/810 045
银行格式	1.33	0.00

舍入格式　　　1.33　　　0.00

对于长格式而言，最后的有效数字不一定正确，但是在计算机内部存储的二进制数是十分精确的。对于长或短格式，如果矩阵的最大元素比 1 000 大或比 0.001 小，则输出时，将自动加入一个长度因数，也就是比例因子，如：

```
x=[12 345 0.000 9]
```

输出结果：

```
x=1. 0e+004 *
    1.234 5   0.000 0
```

即表示 $x=10^4*[1.2345\ \ 0.000]$，其中 10^4 为比例因子。

“＋”格式是显示大矩阵元素为正或负的一种紧凑方法，“＋”、“－”和空格分别表示正、负和零元素。

从 MATLAB 工作空间的 Options 菜单中，可以方便地进行格式选择。在选定了某种格式后，则所有的结果输出都采用此格式，除非用 format 命令特别定义。

选择工作空间中菜单项 Options 中的子菜单项|Numeric Format(数字显示格式)，就可以设置 MATLAB 下数据结果的显示格式，用户可以从中选择 Short(默认的简洁格式)和 Long(高精度格式)等，通过对该选项的设置，用户就可以得到期望的显示格式了。各种显示格式的设定还可以通过 MATLAB 命令 format 来完成，该命令的用法如表 2－2 所示。

表 2－2　format 命令的用法说明

MATLAB 命令	数值范围	注　释
format long	35.833 333 333 333 34	长格式
format short e	3.583 3e+01	短格式 e 方式
format long e	3.583 333 333 333 334e+01	长格式 e 方式
format hex	4041eaaaaaaaaaab	16 进制格式
format bank	35.83	银行格式
format +	+	紧密格式，用“＋”、“－”或空格来示意地表示数字
format rat	215/6	有理格式
format short	35.833 3	短格式

2.1.5　变量的命名

和其它任意一种计算机高级语言一样，MATLAB 语言中对变量的命名有一套规则，如表 2－3 所示。

表 2－3　MATLAB 语言中变量的命名规则

命名规则	注　释
变量名对大小写敏感	fruit，Fruit，FrUit 以及 FRUIT 在 MATLAB 语言中是不同的变量
每个变量名最多可包含 19 个字符	
变量名的首字符必须是字母	标点符号不能出现在变量名中

2.1.6 几条小技巧

(1) 符号%在 MATLAB 语言中作为注释符，如下例所示：

```
apples = 4      % Number of apples
apples =4
```

(2) 几条语句可以出现在同一行中，只要用分号或逗号将它们分割开来。如果在语句末尾是分号“;”，则说明除了这一条命令外还有下一条命令等待输入，MATLAB 这时将不给出运行的中间结果。当所有命令输入完毕后，直接打回车键，则 MATLAB 将给出最终的运行结果。例如：

```
apples = 4 , bananas = 6; cantaloupes = 2
apples = 4
cantaloupes =2
```

如果一条表达式很长，一行放不下，则键入“…”后回车，即可在下一行继续输入。注意在“…”前要留有空格。例如：

```
S=1-1/2+1/3-1/4+1/5-1/6   ...
-1/8+1/9-1/10
```

(3) 在任意状态只要键入 Ctrl-C，即可终止 MATLAB 程序。

(4) 在 MATLAB 的工作空间中键入命令 quit，即可退出 MATLAB。

2.1.7 文件管理命令

表 2-4 列出了 MATLAB 中提供的一些文件管理命令。

表 2-4 文件管理命令

命 令	注 释
what	列出当前目录下所有的 M 文件
dir	列出当前目录下所有的文件
ls	与 dir 命令相同
type test	在命令窗口中显示文件 test.m
delete test	删除文件 test.m
cd path	进入目录
chdir path	进入目录
cd	显示当前目录
chdir	显示当前目录
pwd	显示当前目录
which test	显示文件 test.m 所在的路径

2.2 MATLAB 语言的科学计算特性

2.2.1 常用的数学函数

MATLAB 提供了丰富的数学函数，用户针对自己的不同要求，可以方便地调用函数，大大减少了工作量。表 2-5 给出 MATLAB 中的一部分常用数学函数。

表 2-5 MATLAB 中的部分常用数学函数

MATLAB 语言中的数学函数	注 释
abs(x)	求绝对值或复数的模
acos(x)	反余弦函数
acosh(x)	反双曲余弦
angle(x)	求复数的角度
asin(x)	反正弦函数
asinh(x)	反双曲正弦
atan(x)	反正切函数
atan2(x,y)	第四象限的反正切
atanh(x)	反双曲正切
conj(x)	求共轭复数
cos(x)	余弦函数
cosh(x)	双曲余弦
exp(x)	指数函数
fix(x)	朝零方向取整
floor(x)	朝负无穷方向取整
imag(x)	求复数的虚部
log(x)	自然对数
log10(x)	常用对数
real(x)	求复数的实部
rem(x,y)	除后余数
round(x)	四舍五入到最接近的整数
sign(x)	符号函数
sin(x)	正弦函数
sinh(x)	双曲正弦
sqrt(x)	开平方
tan(x)	正切函数

2.2.2 复数与复数矩阵

在 MATLAB 中，复数单位为 i=sqrt(-1)，其值在工作空间中都显示为 0+1.0000i。复数可由下面的语句生成：

z=a+b * i(可简写成 z=a+bi)

或 z=r * exp(i * 0)(也可简写成 z=r * exp(0i)

其中 0 为复数幅角的弧度数，r 为复数的模。

有两个方便的方法来输入复数矩阵，如：

A=[12 ;34]+i * [56;78]

和 A=[1+5*i 2 +6*i;3+7*i 4+8*i]

两式具有相同的结果。当复数作为矩阵元素时，复数内不能留有空格，如 1+5*i，若在加号前有空格，就会被认为是两个分开的数。事实上任何矩阵的元素内都不能有空格，否则会被 MATLAB 认为是两个元素而出错。

2.3 MATLAB 的在线帮助功能

2.3.1 帮助命令(help)

MATLAB 提供了大量的函数和命令，如果想记住所有的函数及其调用格式几乎是不可能的。为此，MATLAB 提供了在线帮助的功能，通过这一功能，用户可以容易地获得对想查询的各个函数的有关信息，在线查询可以由 help 命令来获得。

在 MATLAB 的工作空间中直接键入 help，即可得到：

```
HELP topics:
toolbox\local          — Local function library.
matlab\datafun         — Data analysis and Fourier transform functions.
matlab\elfun           — Elementary math functions.
matlab\elmat           — Elementary matrices and matrix manipulation.
matlab\funfun          — Function functions - nonlinear numerical methods.
matlab\general         — General purpose commands.
matlab\color           — Color control and lighting model functions.
matlab\graphics        — General purpose graphics functions.
matlab\iofun           — Low -level file I/O functions.
matlab\lang            — Language constructs and debugging.
matlab\matfun          — Matrix functions - numerical linear algebra.
matlab\ops             — Operators and special characters.
matlab\plotxy          — Two dimensional graphics.
matlab\plotxyz         — Three dimensional graphics.
matlab\polyfun         — Polynomial and interpolation functions.
matlab\sounds          — Sound processing functions.
matlab\sparfun         — Sparse matrix functions.
matlab\specfun         — Specialized math functions.
matlab\specmat         — Specialized matrices.
matlab\strfun          — Character string functions.
matlab\dde             — DDE Toolbox.
matlab\demos           — The MATLAB Expo and other demonstrations.
toolbox\control        — Control System Toolbox.
toolbox\hispec         — Hi -Spec Toolbox
simulink\simulink      — SIMULINK model analysis and construction functions.
simulink\simdemos      — SIMULINK demonstrations and samples.
simulink\blocks        — SIMULINK block library.
```

```
simulink\sb2sl          — SystemBuild 3.0 model import into SIMULINK.
toolbox\ident           — System Identification Toolbox.
toolbox\signal          — Signal Processing Toolbox.
toolbox\wintools        — GUI tools for MATLAB for MS Windows.
nnet\nnet               — Neural Network Toolbox.
nnet\nndemos            — Neural Network Demonstrations and Applications.
toolbox\optim           — Optimization Toolbox.
For more help on directory/topic, type "help topic".
```

如果要对某一命令或函数进行查询，直接在 help 后跟上该命令或函数名即可。例如：

```
help sqrt
SQRT Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.
    See also SQRTM.
```

2.3.2 查找命令(lookfor)

MATLAB 还提供了关键词查询命令 lookfor，例如若想查询和 complex(复数)有关的命令和函数，则可以在 MATLAB 的工作空间中键入 lookfor complex，并得到如下结果：

```
   lookfor complex
CPLXPAIR Sort numbers into complex conjugate pairs.
CONJ     Complex conjugate.
IMAG     Complex imaginary part.
REAL     Complex real part.
CDF2RDF Complex diagonal form to real block diagonal form.
RSF2CSF Real block diagonal form to complex diagonal form.
CPLXDEMO Maps of functions of a complex variable.
CPLXGRID Polar coordinate complex grid.
CPLXMAP Plot a function of a complex variable.
GRAFCPLX Demonstrates complex function plots in MATLAB.
DSORT    Sort complex discrete eigenvalues in descending order.
ESORT    Sort complex continuous eigenvalues in descending order.
LOGM2    LOGM2(X)  is the matrix natural logarithm of X .   Complex
LOGM2    LOGM2(X)  is the matrix natural logarithm of X .   Complex
PHASE    Computes the phase of a complex vector
CCEPS Complex cepstrum.
```

2.3.3 从菜单上获取帮助

MATLAB 还提供了 Windows 下的查询方法，这和一般 Windows 程序的联机帮助系统是统一的。例如用户选中 MATLAB 工作空间中 Help 菜单中的 Tables of Contents ... 子菜单选项，则将出现如图2-3所示的帮助界面。可以看出，帮助界面里给出的帮助信息就像一本帮助手册的目录一样，可以进一步查询的条目都用带下划线文字给出，用户可以用鼠标点中其中的条目来进行进一步的查询。除了顺序地查询方式以外，用户还可以点中图2-3所示界面中的 Search(查询)按钮来查询自己所给出的关键词，这时候将得到如图2-4所示的图形界面，

用户可以在最上面的对话框中输入要查询的内容，然后按下 Show Topic（显示主题）按钮寻找感兴趣的问题。



图 2-3 MATLAB 的在线帮助系统

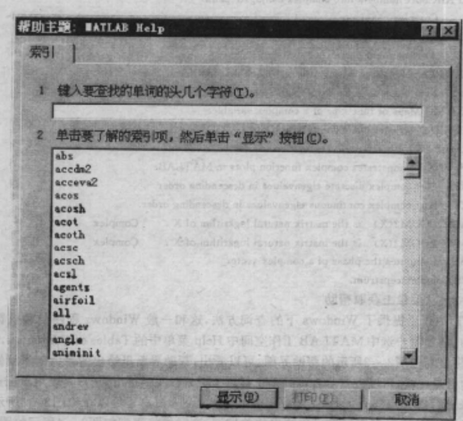


图 2-4 MATLAB 在线帮助系统的查询界面

第三章　MATLAB 语言程序设计

MATLAB 提供了丰富的编程语句结构和实用函数，使用 MATLAB 编程运算与人进行科学计算的思路和表达方式完全一致，所以学习起来极为方便。MATLAB 大大降低了对使用者的数学基础和计算机语言知识的要求，而且编程效率和计算效率大为提高，还可在计算机上直接输出结果和精美的图形。本章详细介绍 MATLAB 语言的程序设计。

3.1　向量运算

3.1.1　向量的构造

在 MATLAB 中"："是一个重要的字符，如 x＝1:4即产生一个1～4单位增量的行向量：

x＝1 2 3 4

也可以产生单位增量小于1的行向量，方法是把增量放在起始和结尾量的中间，并用冒号分割开来。如：

y＝0:pi/4:pi 即产生一个从0到 π 的行向量，单位增量是 π/4＝0.785 4，所以得

y＝[0　0.785 4　1.570 8　2.356 2　3.141 6]

也可以产生单位增量为负数的行向量，如：z＝7:－1:2，即得

z＝[7　6　5　4　3　2]

符号"："也可以用来产生简易的表格。为了产生纵向表格形式，首先要用它产生行向量，然后进行转置，再利用所得的列向量计算出另一列向量，即可合成有两列的矩阵。例如：

```
x=(0:0.2:3.0)';
exp(-x).*sin(x);
[x y]
```

则得到矩阵

$$
ans=\begin{bmatrix}
0 & 0 \\
0.200\,0 & 0.162\,7 \\
0.400\,0 & 0.261\,0 \\
0.600\,0 & 0.309\,9 \\
0.800\,0 & 0.322\,3 \\
1.000\,0 & 0.309\,6 \\
1.200\,0 & 0.280\,7 \\
1.400\,0 & 0.243\,0 \\
1.600\,0 & 0.201\,8 \\
1.800\,0 & 0.161\,0 \\
2.000\,0 & 0.123\,1 \\
2.200\,0 & 0.089\,6 \\
2.400\,0 & 0.061\,3 \\
2.600\,0 & 0.038\,3 \\
2.800\,0 & 0.020\,4 \\
3.000\,0 & 0.007\,0
\end{bmatrix}
$$

3.1.2 下标

MATLAB的下标具有很重要的功能，可以在对矩阵的行、列子矩阵处理时使用，也可以用来产生向量。使用下标和向量，会使运算更为清晰和方便。单个的矩阵元素可在括号中用下标来表达。例如，已知

$$A=\begin{bmatrix} 1 & 2 & 4 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

其中元素 A(3,3)＝9，A＝(1,3)＝4，A(3,1)＝3等等.若用语句 A(3,3)＝A(1,3)＋A(3,1)，利用原矩阵的元素产生新元素(即为 A(3,1)＋A(1,3)＝7)替代 A 矩阵中第三行第三列的元素 A(3,3)，则产生的新的 A 矩阵为

$$A=\begin{bmatrix} 1 & 2 & 4 \\ 2 & 5 & 8 \\ 3 & 6 & 7 \end{bmatrix}$$

下标可以是一个向量，例如若 x 和 v 都是向量，则 x(v)也是一个向量：[x(v(1)) x(v(2)) …x(v(n))]。对于矩阵来说，向量下标可以将矩阵中邻近或不邻近元素构成一新的子矩阵；假设 A 是一个10×10的矩阵，则 A(1:5,3)指 A 中由前5行对应第三列元素组成的5×1子矩阵。又如 A(1:5,7:10)是前5行对应最后4列组成的5×4子矩阵。

使用":"代替下标，可以表示所有的行或列。如：A(:3)代表第三列元素组成的子矩阵，A(1:5,:)代表由前5行所有元素组成的子矩阵。

对于子矩阵的赋值语句，":"有更明显的优越性。如

A(:,[3,5,10])＝B(:,1:3)

表示将 B 矩阵的前三列，赋值给 A 矩阵的第三、第五和第十列。

通常如果 v 和 w 是具有整数性质的向量，则 A(v,w)通过取出行下标 v 和列下标 w 对应的 A 的元素而形成新的矩阵。于是，A(:,n:－1:1)即为由原来 A 矩阵中取 n 至1负增长的列的元素组成一个新的矩阵，其行数仍为原来 A 矩阵的行数，列数为 n。

如果 v＝2:2:n；w＝[3 1 4 1 6]，此时 A(v,w)是合法的，但并不排除出问题的可能性。

进一步分析，A(:)在赋值语句的右边，表示将 A 的所有元素在一个长的列向量中展成串，如

A＝[1 2;3 4;5 6]，b＝A(:)

则结果为

$$A=\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$b=\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

在赋值语句左边A(:)可以重新组成与刚才的A具有相同阶数的矩阵,这相当于在原来的A没有被清除的情况下,用新的元素置换,实际上起着一种提供格式的作用。例如

A(:)=11:16

即可得

$$A=\begin{bmatrix} 11 & 14 \\ 12 & 15 \\ 13 & 16 \end{bmatrix}$$

这时A矩阵已被新的内容所取代了,即由11至16,6个新元素取代原来矩阵中的元素,重新组成3×2矩阵。

3.1.3 具有0—1向量的下标

从关系运算中建立的0—1向量可以用于参考建立子矩阵,假设A是一个m×n矩阵,L是一个m维的0—1向量,则A(L,:)将给出L中非零元素所对应的A的行元素所组成的子矩阵。即

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad L=\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

A(L,:)

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

3.1.4 空矩阵

语句X=[]将一个0×0的矩阵赋给X,使用这个矩阵,不会引起出错情况。X=[]与clear X不同,clear是将X清除出工作空间,而空矩阵则存在于工作空间,只是具有"空尺寸"而已,当用size测试表明其为空矩阵时,exist可以测出其确实存在。

如果给出空矩阵,确定的矩阵函数如det( ),cond( ),prod( ),sum( )等会返回一个值。例如:当给定空矩阵时,prod( )、det( )、和sum( )将分别返回1、1和0。

空矩阵在数学上讲它本身是空的,我们不能对其进行一般意义上的处理,但可以发现这个概念的确是很有用的。

3.2 关系和逻辑运算

3.2.1 关系运算

MATLAB中有6个关系操作符,如表3-1所示。

表3-1 关系操作符

关系操作符	注 释
<	小于
<=	小于或等于
>	大于
>=	大于或等于
==	等于
~=	不等于

比较两个元素的大小时，结果是1表明为真，结果是0表明为假。例如2＋2～＝4(意为2加2不等于4)结果是0表明为假。

函数 find( )在关系运算中很有用，它可以在0—1矩阵中寻找一个非零元素或在其它矩阵中找出一些满足一定条件的数据元素。假如 y 是一个向量 ，find(y＜3.0)则返回一个向量，它表示 y 中满足此条件元素的位置。例如，键入下列命令：

```
i=find(y>3.0)
y(i)=10*ones(i)
```

上面的程序段表示用10来代替 y 中所有大于3的元素。

3.2.2 逻辑运算

MATLAB 语言中有三种逻辑运算，如表3－2所示。它们通常用于元素或0—1矩阵的逻辑运算。

表 3－2 MATLAB 中的逻辑运算

逻辑运算	注 释
&	与
|	或
～	非

“&”和“|”操作符可以比较两个标量或两个同阶矩阵(或数组)，对于矩阵而言，逻辑运算符是作用于矩阵中的元素。例如 A，B 是0—1矩阵，那么 A&B 是表示 A、B 中相应元素间进行逻辑与运算所得到的另一个0—1矩阵，逻辑运算结果信息也用0和1表示，逻辑操作符认定任何非零元素都表示为真。给出1为真，0为假。

非(或逻辑非)是一元操作符，当 A 非零时“～A”返回信息为0，当 A 为零时返回信息为1。因而就有 p|(～p)返回值为1；p&(～p)返回值为0。

3.2.3 有关的关系和逻辑运算函数

除了上面介绍的关系和逻辑运算符之外，MATLAB 中还提供了一些关系和逻辑运算函数，如表3－3所示。

表 3－3 其它的关系和逻辑运算符号

函 数	注 释
xor(x,y)	异或
any(x)	向量 x 中的任一元素非零，返回1
all(x)	向量 x 中的每一个元素非零，返回1
isnan(x)	当 x 是 NaN 时，返回1
isinf(x)	当 x 是 Infs 时，返回1
finite(x)	当 x 属于(－∞，＋∞)时，返回1，而当 x=NaN 时，返回零

any 和 all 函数在连接操作时很有用，如果 x 是0—1向量，且 x 中任一个元素不为零，any(x)返回1，否则返回0，当 x 的所有元素非零时，all(x)函数返回1，这些函数在 if 语句中特别有

用，如：

```
if all(A<.5)
do something
end
```

对于矩阵，any 和 all 命令按列对其处理并返回带有处理列所得结果的一个行向量。双函数，如 any(any(A)) 总是将矩阵简化成一个标量条件。

3.3 矩阵运算

矩阵运算是 MATLAB 的基础，MATLAB 语言对矩阵进行处理的能力是极强的。

3.3.1 矩阵转置

矩阵的转置用符号"'"来表示和实现。

例如：A=[1 2 3; 4 5 6; 7 8 9]，转置矩阵 B=A'，则

$$B=\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$$

再如直接对向量进行转置运算，则[-1 0 2]'的结果为

$$ans=\begin{bmatrix}-1\\ 0\\ 2\end{bmatrix}$$

如果 z 是复数矩阵，则 z'为它们的复数共轭转置。非共轭转置使用 z.'或 conj(z')求得。

3.3.2 矩阵加和减

矩阵的加减运算使用的是"+"和"-"运算符。而矩阵必须具有相同阶数才可进行加、减运算。例如 A 是3 ×3 矩阵，X 是3×1 矩阵，就不能进行 A+X 运算。若

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}, \qquad B=\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$$

则 C=A+B 是可行的，值为

$$C=\begin{bmatrix}2 & 6 & 10\\ 6 & 10 & 14\\ 10 & 14 & 18\end{bmatrix}$$

可见矩阵的加减运算是其对应元素的加减运算。

如果运算对象是一个标量即 1×1 矩阵，它可以和其它不同阶数的矩阵进行加减运算，如：x=[-1 0 2]'; y=x-1，则

$$y=\begin{bmatrix}-2\\ -1\\ 1\end{bmatrix}$$

3.3.3 矩阵乘法

矩阵乘法用"*"表示，当两矩阵中前一矩阵的列数和后一矩阵的行数相同时，可以进行乘法

运算，这与数学上的形式是一致的，两个相同维数向量的内积(数学上称为点积，标量乘)也可以用这种乘法实现。

例如：x=[-1 0 2]'，y=[-2 -1 1]'，则运算 x' * y 和 y' * x 都将得到结果：ans=4

MATLAB 中对于计算向量的叉乘(矢积)没有特别地给出，但是通过编写 M 文件可以很容易地计算。

在 MATLAB 中还可进行矩阵和标量相乘，标量可以是乘数也可以是被乘数。矩阵和标量相乘是进行矩阵中的每个元素都与此标量相乘的运算。

例如计算 pi * x，其中 x=[-1 0 2]，则

$$\text{ans}=\begin{bmatrix} -3.1416 \\ 0.0000 \\ 6.2832 \end{bmatrix}$$

3.3.4 矩阵除法

在 MATLAB 中用两种矩阵除法符号 "\\"和"/"分别表示左除和右除。如果 A 矩阵是非奇异方阵，则 A\B 和 B/A 运算可以实现。A\B 等效于 A 的逆左乘 B 矩阵，也就是 inv(A) * B，而 B/A 等效于 A 矩阵的逆右乘 B 矩阵。

通常 X=A\B 是 A * X=B 的解，X=A/B 是 X * A=B 的解。一般情况下 A\B 不等于 A/B。

3.3.5 矩阵的乘方

A ^ P表示 A 的 P 次方。如果 A 是一个方阵，P 是一个标量，且 P 是大于1的整数，则 A 的 P 次幂即为 A 自乘 P 次。如果 P 不是整数，则计算涉及特征值和特征向量的问题。例如若 [V,D]=eig(A)，则

A ^ P=V * D. ^ P/V

其中 V 和 D 分别为矩阵 A 的特征向量矩阵和特征值矩阵。

如果 P 是矩阵而 A 是标量，以及 A、P 都是矩阵，则 A ^ P 都是不成立的。

3.3.6 矩阵的超越函数

在 MATLAB 中 exp、sqrt 等命令也可作用到矩阵上，但这种运算只是定义在矩阵的单个元素上，即分别对矩阵的每一个元素进行计算。

超越数学函数可以在函数后加上 m 而成为矩阵的超越函数，例如 expm(A)、sqrtm(A)、logm(A)分别为矩阵指数、矩阵开方和矩阵对数。这种运算用 funm(A ，'exp')等也可以实现，但它们所用的算法不同。

矩阵的超越函数要求运算矩阵必须为方阵。

3.3.7 建立矩阵的函数

用以建立矩阵的函数有以下几种：

(1) eye(size(A))产生与 A 矩阵同阶的单位矩阵。

(2) zeros 和 ones 产生0和1矩阵。

(3) rand 产生随机元素的矩阵。

(4) 函数 diag( )、triu( )和 tril( )分别提供对角、上三角和下三角矩阵。

(5) 函数 size( )显示一个包含两个元素的向量：矩阵的行数和列数。函数 length( )返回向量的长度或矩阵行数列数的最大者：length(x)=max(size(x))。

3.4 矩阵函数

MATLAB 的数学处理能力很大一部分是由它的矩阵函数扩展而来的。MATLAB 矩阵函数主要有三角分解、正交变换、奇异值分解和特征值等几方面的内容。

3.4.1 三角分解

基本的分解是将一个方阵表示成两个基本三角阵的乘积，其中一个三角阵为上三角阵，另一个为下三角阵。这种分解通常被称为"LU"分解，这里使用的算法是高斯变量消去法。分解的因数从 lu 函数中得到，利用这种分解，求逆(inv)可得到逆矩阵，求 det 可得到行列式的值。它是求解线性方程的基础，也是方阵"\"或"/"两种矩阵除法的基础。例如对

$$A=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

进行 LU 分解，可用下列语句来实现：

```
[L,U]=lu(A)
```

得到结果为

$$L=\begin{bmatrix} 0.1429 & 1.0000 & 0 \\ 0.5714 & 0.5000 & 1.0000 \\ 1.0 & 0 & 0 \end{bmatrix}, \quad U=\begin{bmatrix} 7.0000 & 8.0000 & 0.0000 \\ 0 & 0.8571 & 3.0000 \\ 0 & 0 & 4.5000 \end{bmatrix}$$

注：L 是转换了的对角线为1的下三角矩阵，U 是上三角矩阵，逆运算可证明 LU 分解的正确性，由 L，U 生成 A：

```
L*U
```

$$ans=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

求逆得

$$X=inv(A)=\begin{bmatrix} -1.7778 & 0.8889 & -0.1111 \\ 1.5556 & -0.7778 & 0.2222 \\ -0.1111 & 0.2222 & -0.1111 \end{bmatrix}$$

从三角分解转换的求逆是精确的：X=inv(U)*inv(L) 取得了与上式直接求逆相同的值。行列式的值为

$$d=det(A)=27$$

由三角分解的因数矩阵计算行列式的值

$$d=det(L)*det(U)=27.0000$$

注意到两种 d 的显示格式不一样，这是因为当 MATLAB 做 det(A)运算时，A 的所有元素都是以整数形式给出，运算中不出现小数，故结果也是整数；而用分解因数矩阵计算 d 时，因数 L、U 中的元素已非全整数，故这样产生的 d 也不是整数。

3.4.2 正交变换

"QR"分解对方阵和长方矩阵都很有用，它表示为正交矩阵和上三角矩阵的乘积。例如：

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\\ 10 & 11 & 12\end{bmatrix}$$

是一个秩亏矩阵，中间列是其它两列的平均，秩和亏损值由“QR”分解得出。

[Q,R]=qr(A)

$$Q=\begin{bmatrix}-0.0776 & -0.8331 & 0.5444 & 0.0005\\ -0.3105 & -0.4512 & -0.7709 & 0.0351\\ -0.5433 & -0.0694 & 0.0913 & -0.8317\\ -0.7762 & 0.3124 & 0.3178 & 0.4461\end{bmatrix}$$

$$R=\begin{bmatrix}-12.8841 & -14.5916 & -16.2992\\ 0 & -1.0413 & -2.0826\\ 0 & 0 & 0.0000\\ 0 & 0 & 0\end{bmatrix}$$

容易验证 Q * R 的值就是原矩阵 A.观察 R 矩阵的三角结构，其对角线的下半部全是0，在对角线上 R(3,3)元素为零值.说明 R 与原来 A 矩阵不是满秩的。QR 分解可以解决方程数多于未知数的线性系统问题。

3.4.3 奇异值分解

在 MATLAB 中，奇异值分解函数 svd( )的调用格式为：[U,S,V]=svd(A)

在奇异值分解中产生三个因数矩阵 U、S 和 V，并且：A=U * S * V'

U 矩阵和 V 矩阵是正交矩阵，S 矩阵是对角矩阵，而且 svd(A)恰好返回 S 的对角元素，就是 A 的奇异值。

其它几种函数也可被进行奇异值分解，这包括广义逆矩阵 pivn(A)，秩 rank(A)，欧几里德矩阵范数 norm(A,2)和条件数 cond(A)。

3.4.4 特征值

如果 A 是 n×n 矩阵，n 个 λ 值满足式 Ax=λx，则 λ 为 A 的特征值，x 为 A 的特征向量。计算特征值使用函数 eig(A)，以列向量形式返回特征值。如果 A 是实对称矩阵，特征值为实数，如果不对称，特征值常为复数，例如

$$A=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

eig(A)产生

$$ans=\begin{bmatrix}0.0000+1.0000i\\ 1.0000-1.0000i\end{bmatrix}$$

求解特征值和特征向量可以用双赋值语句得到：[X,D]=eig(A)

D 的对角元素是特征值，X 为矩阵，它的列是相应的特征向量，以使得 A * X=X * D。

3.4.5 秩和条件

在 MATLAB 中有关秩与条件的主要函数如表3-4所示。

表 3-4

函数	注释
cond	2-范数条件数
norm	矩阵范数
rank	秩
rcond	条件估计

MATLAB 中计算秩的方法有:rref(A)法、对于非方阵 A 的 A\B 有 orth(A)及 null(A)方法、广义逆矩阵 pinv(A)方法等。由于采用三种不同舍入规则,三种不同算法有可能对同一矩阵得到三个不同的值。

利用 rref(A),A 的秩为非零行的数目。rref 算法是三种定秩算法中最简单的一个,但是最不可靠;pinv(A)是基于奇异值的算法,它消耗时间最多,但也最为可靠;另外在计算满秩时,rank(A) 也十分有用。

3.5 控制语句

3.5.1 MATLAB 的循环语句结构

MATLAB 与其它计算机语言一样,有控制流语句。这使得 MATLAB 语言的编程显得十分灵活。

一、for 循环

for 循环完成一条语句或一组语句在一定情况下的反复使用,其重复的次数是预先设定的。例如:

```
for i=1:n,x(i)=0;end
```

x 前 n 个元素被依次赋零值。如果 n<1,结构上仍然是合法,但内部并不运行。如果 x 中不存在或少于 n 个元素,则缺少的会被自动附加上去。

循环语句书写成锯齿形是为了增加可读性。例如:

```
for i=1:m
   for j =1:n
      A(i,j)=1/(i+j-1);
   end
end
A
```

语句内的分号终止输入部分并防止对 A 中间结果的打印输出。非常重要的是,for 语句一定要有 end 命令作为结束,否则下面的输入都被它接受为 for 循环的内容。循环结束后,键入 A 则显示最终结果。

二、while 循环

MATLAB 中的 while 循环语句是控制一个或一组语句在一个逻辑条件下重复预先不确定的次数。例如:

```
n=1;
```

```
while prod(1:n)<1.e100,n=n+1;end
n
```

3.5.2 MATLAB 的条件转移语句结构

最简单的条件转移语句结构如下：

```
if    expression
      conmmands
end
```

复杂的条件转移语句结构如下：

```
if    expression1
          conmmands evaluated if expression1 is True
elseif expression2
          conmmands evaluated if expression2 is True
elseif expression3
          conmmands evaluated if expression3 is True
elseif expression4
          conmmands evaluated if expression4 is True
elseif …
          ·
          ·
          ·
else
          conmmands evaluated if no other expression is True
end
```

3.6 数值分析

用户经常需要解析地求出函数的积分、微分或者是确定函数中特定的点，但当函数比较复杂时就会遇到困难。这时可以利用数值分析的方法来近似求解，MATLAB 提供了一些用于数值分析的工具。

3.6.1 函数图形绘制

当需要对一个函数进行分析时，有必要首先绘制出函数的图形。在此，可以利用函数 fplot( )来绘制函数的图形。例如：

```
fplot('humps',[0,2])
```

则产生图形如图3－1所示。其中'humps'是 MATLAB 中的一个 M 文件函数，如下所示：

```
function y = humps(x)
%HUMPS      A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
%           HUMPS(X) is a function with strong maxima near x = .3 and x = .9.
%           See QUADDEMO, ZERODEMO and FPLOTDEMO.
%           Copyright (c) 1984-94 by The MathWorks, Inc.
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

对于一些简单的函数，例如 $y=2e^{-x}\sin(x)$，也可以不编写 M 文件函数，直接调用 fplot( ) 函数，绘制函数的图形。具体实现如下：

```
f='2*exp(-x).*sin(x)';
fplot(f,[0,8])
```

上面程序段产生的图形如图3－2所示。

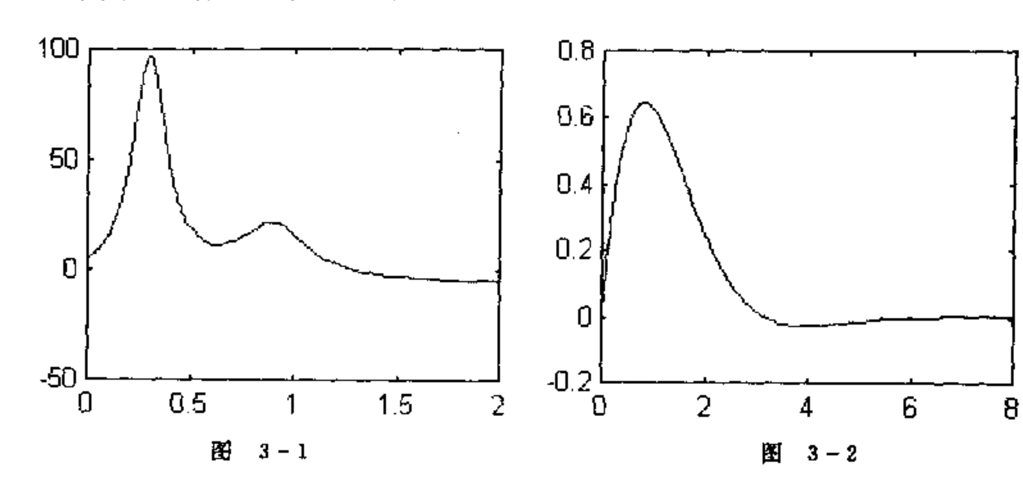


图 3－1　　　　图 3－2

3.6.2 求极值

在很多实际应用中，都需要求出函数的极点。MATLAB 提供了两个函数：fmin( )和 fmins( )来求函数的极点。其中 fmin( )用于一维函数，fmins( )用于多维函数。以图3－2中的函数为例，若要求其最小值，则键入下列命令：

```
fn='2*exp(-x)*sin(x)';          %定义函数
xmin=fmin(fn,2,5)               %在范围2<x<5内求最小值
```

得到

```
xmin =3.927 0
```

若要求最大值，则键入命令：

```
fx='-2*exp(-x)*sin(x)';         %定义函数，注意函数前的负号
xmax=fmin(fx,0,3)               %在范围0<x<3内求最大值
```

得到

```
xmax =0.785 4
```

3.6.3 求零点

求函数的零点，在实际应用中也会经常用到，MATLAB 提供了函数 fzero( )，可以方便地求得零点。以图3－1中的函数为例，发现函数的零点大致在 x=1.2附近，这时就可以用函数 fzero( )来求零点如下：

```
xzero=fzero('humps',1.2)        %在 x=1.2附近求零点
```

得到结果如下：

```
xzero =1. 299 5
```

3.6.4 积分

MATLAB 提供了函数 trapz( )、quad( )、quad8( )来求函数积分的数值解，在二维情况

下，求积分实质上是求函数曲线与坐标横轴之间所夹的封闭图形的面积。例如：

```
x=-1:.07:2;
y=humps(x);
area=trapz(x,y)
```

得到结果如下：

```
area =26.624 3
```

3.7 二维图形的绘制

MATLAB 风靡全球的另一个重要原因是因为它提供了方便快捷的绘图功能。在MATLAB等软件出现之前，如果想在自己程序中产生一个图形是相当麻烦的。例如如果用户想在自己的C语言程序中绘制一个图形，首先需要对绘图的数据进行预处理，找出这些数据的最大值和最小值，根据它们自动地计算出坐标轴的范围，然后再调用一些绘图命令库函数来将图形在屏幕上显示出来。这样做显然将耗费程序设计者大量的精力，而且绘制的图形效果往往取决于设计者的编程经验，绘制出的图形不一定令人满意。MATLAB 为用户提供了丰富的图形绘制功能，相信用户在 MATLAB 环境下再也不会为绘图而担忧了。

在这一节，我们举几个实际的例子来介绍二维图形绘制。

3.7.1 二维图形绘制入门

例1 如果用户想绘制出一个周期内的正弦曲线，那么在MATLAB的工作空间中键入下列命令：

```
x = linspace(0,2*pi,30);
y = sin(x);
plot(x,y)
```

则可绘制出所需的正弦曲线，如图3-3所示。

有时在实际工作中，需要将不同的曲线绘制在同一个图形窗口中以便于比较，在MATLAB中允许在一个绘图窗口中同时绘制多条曲线，例如在工作空间中继续键入命令：

```
z = cos(x) ;
plot(x,y,x,z)
```

则在同一个图形窗口中得到一个周期内的正弦和余弦曲线，如图3-4所示。

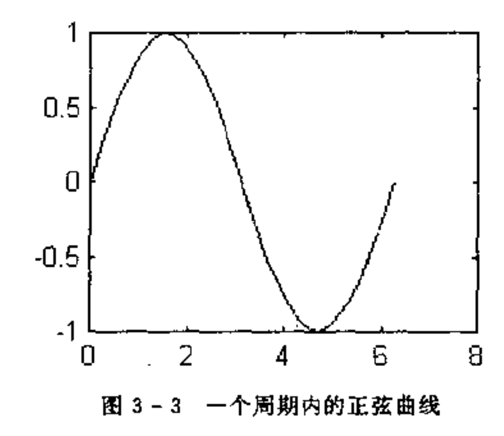


图 3-3 一个周期内的正弦曲线

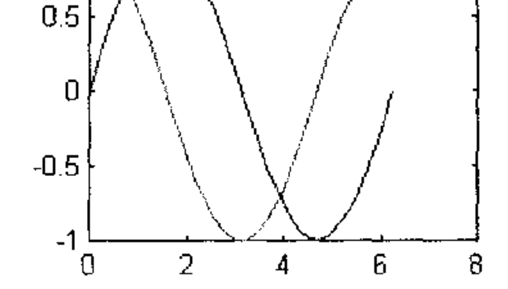


图 3-4 一个周期内的正弦和余弦曲线

3.7.2 线型、记号和颜色

当多条曲线绘制在同一个图形窗口中时，可以用不同的线型和颜色将它们方便地区分开来。MATLAB 提供了一些确定曲线线型和颜色的符号选项，这些符号选项如表3－5所示。

表 3－5 线型、记号、颜色各选项的含义

线型、记号选项	含　义	颜色选项	含　义
·	用点号绘制各个数据点	y	黄色
ｏ	用圆圈绘制各个数据点	m	洋红色
×	用叉号绘制各个数据点	c	蓝绿色
＋	用加号绘制各个数据点	r	红色
*	用星号绘制各个数据点	g	绿色
—	实线	b	蓝色
:	点线	w	白色
—.	点划线	k	黑色
——	虚线		

例2 用不同的线型、记号、颜色在同一个图形窗口中绘制出两条曲线。

在 MATLAB 的工作空间中键入如下命令：

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
plot(x,y,'g:',x,z,'r--',x,y,'w0',x,z,'c+')
```

则可绘制出不同线型、不同颜色、不同记号的曲线，如图3－5所示。

当然，在本书中，不同的颜色没有办法体现出来，在彩色显示器上，这一点将得到清楚的反映。

3.7.3 二维图形的修饰

绘制完曲线后，MATLAB 还提供了一些特殊绘图函数来进一步修饰画出的图形，例如给图形加上网格(grid)、对坐标轴进行标准说明(xlabel、ylabel)、给图形加上图题(title)，此外，还可以在图形的任意位置加上说明性的文本(text)。下面用一个简单的例子来说明这些特殊函数的用法。

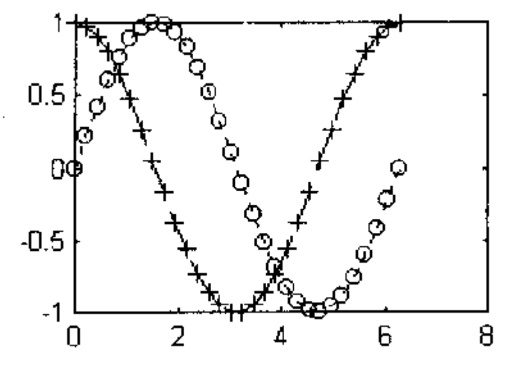


图 3－5 不同线型、不同颜色、不同记号的曲线

例3 在 MATLAB 的工作空间中键入下列命令：

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
```

```
plot(x,y,x,z)
grid
xlabel('Independent Variable X')
ylabel('Dependent Variable Y and Z')
title('Sine and Cosine Curve'])
```

即可得到加上了标注和网格的图形，如图3-6所示。

如果要在图形中的任意位置上加上说明性的文本字符串，可以使用命令 text(x,y,'string')，其中坐标(x,y)代表字符串'string'的左边缘所在点的位置。例如要在图3-6中坐标为(2.5,0.7)的位置加上文本'sin(x)'，则在 MATLAB 工作空间中继续键入下列命令：

```
text(2.5,0.7,'sin(x)')
```

即可得到如图3-7所示的图形。

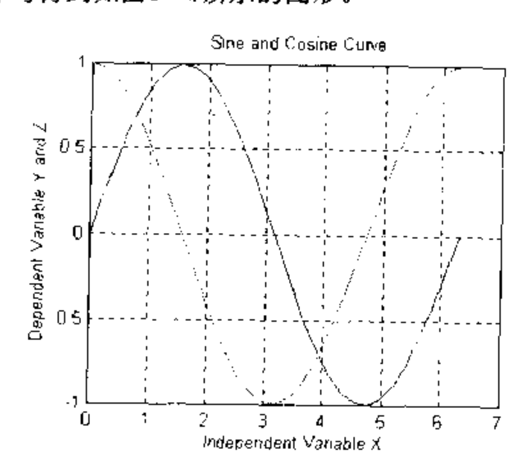


图 3-6 加上标注和网格的图形

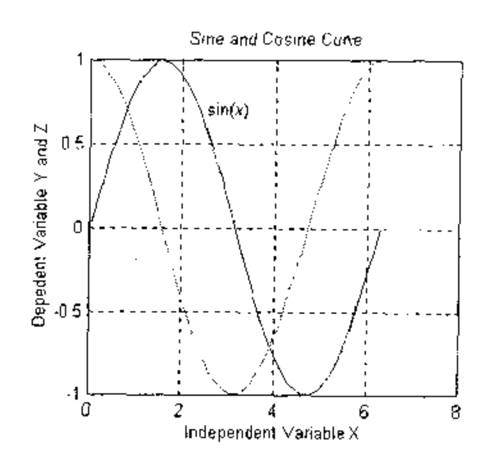


图 3-7 在图形中坐标为(2.5,0.7)处加上文本标注

用 text 命令可以在图形中的任意位置加上文本说明，但是必须知道其位置坐标。而利用另一个函数 gtext，则可以用鼠标来对要添加的文本字符串定位。在 MATLAB 的工作空间中继续键入下列命令：

```
gtext('cos(x)')
```

那么在图中，将会出现一个十字叉，用鼠标拖动它到添加文本的位置，单击鼠标，gtext 命令中的文本字符串就自动添加到图中，如图3-8所示。

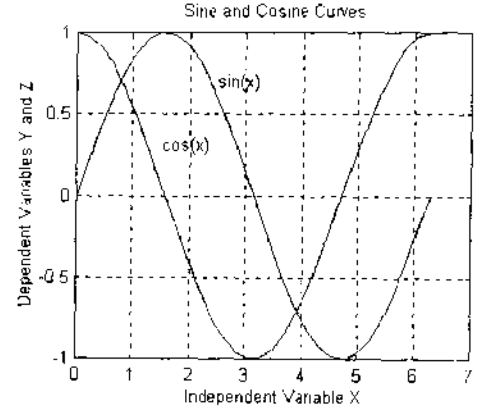


图 3-8 用鼠标定位添加文本说明的图形

3.7.4 二维图形坐标轴的修改

MATLAB 可以自动根据曲线数据的范围选择合适的坐标系，从而使得曲线能够尽可能清晰地显示出来，所以一般情况下用户不必担心图形坐标的选择。但是如果用户对图中横坐标和纵坐标不太满意，还可以利用函数 axis( )来对坐标轴进行修改。axis( )的功能非常丰富，在此仅列出它的一些常用用法，如表3-6所示。

表 3-6 axis( )的常用功能

命令形式	注 释
axis([xmin xmax ymin ymax])	按照用户给出的 x 轴和 y 轴的最大、最小值选择坐标系
axis auto	自动设置:xmin=min(x);xmax=max(x); ymin=min(y);ymax=max(y)。
axis('auto')	自动设置:xmin=min(x);xmax=max(x); ymin=min(y);ymax=max(y)。
axis(axis)	将当前的坐标范围值固定下来,使得 hold 开关打开,接下来的图形都采用当前坐标范围值
axis xy	使用迪卡尔坐标系
axis('xy')	使用迪卡尔坐标系
axis ij	使用 matrix 坐标系,即:坐标原点在左上方,横坐标从左向右增大,纵坐标从上向下增大
axis('ij')	使用 matrix 坐标系,即:坐标原点在左上方,横坐标从左向右增大,纵坐标从上向下增大
axis square	将当前图形设置为正方形图形
axis('square')	将当前图形设置为正方形图形
axis equal	将横纵坐标的单位刻度设置为相等
axis('equal')	将横纵坐标的单位刻度设置为相等
axis normal	关闭 axis equal 和 axis square
axis('normal')	关闭 axis equal 和 axis square
axis off	关闭网格、横纵坐标的注释(label),但保留用 text( )命令和 gtext( )命令添加的文本说明
axis('off')	关闭网格、横纵坐标的注释(label),但保留用 text( )命令和 gtext( )命令添加的文本说明
axis on	打开网格、横纵坐标的注释(label)
axis('on')	打开网格、横纵坐标的注释(label)

3.7.5 图形打印

MATLAB 中图形的打印既可以通过菜单(MENU)来操作,还可以在工作空间中直接键入命令。选用 File 菜单项中的 Print 项,就可以打印图形,当然还可以通过 Print Setup 项和 Page Setup 项来设置参数。在直接用命令 print 之前,必须首先激活图形窗口,可以用鼠标单击图形窗口使之成为当前活动窗口,也可以键入命令 figure(n),然后就可以使用图形打印命令 print。

此外,可以利用命令 orient 来对图形打印的位置进行设置。打印位置的设置有三种模式。缺省模式是 portrait,在这种模式下,图形打印在纸张的中心。

3.7.6 图形窗口的分割

在实际工作中，有时需要在同一个图形窗口中绘制多个图形，这时候就需要对图形窗口进行分割。分割图形窗口的工作是由函数 subplot( )来设置的。该函数的调用格式是 subplot(n，m，k)，其中，n、m 分别表示将这个图形窗口分割的行数和列数，而 k 表示要画图部分的代号。

下面给出一个例子来说明图形分割的具体运用。

例4 将图形窗口分割成4份，并分别绘制图形。

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
a=2*sin(x).*cos(x);
b=sin(x)./(cos(x)+eps);
subplot(2,2,1)
plot(x,y),axis([0 2*pi -1 1]),title('sin(x)')
subplot(2,2,2)
subplot(x,z)
plot(x,z),axis([0 2*pi -1 1]),title('cos(x)')
subplot(2,2,3)
plot(x,a),axis([0 2*pi -1 1]),title('2sin(x)cos(x)')
subplot(2,2,4)
plot(x,b),axis([0 2*pi -20 20]),title('sin(x)/cos(x)')
```

得到图形如图3－9所示。

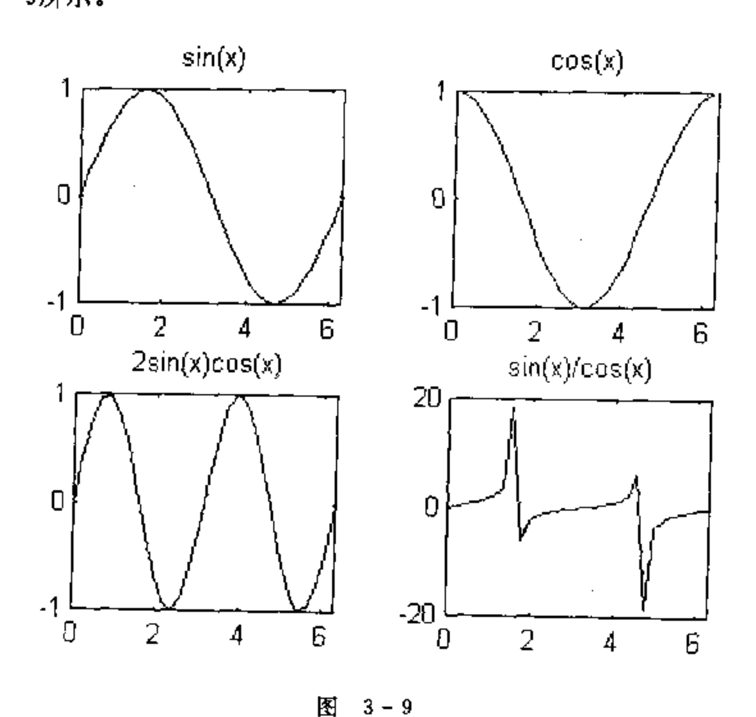


图 3－9

3.7.7 特殊坐标图形

除了前面介绍的二维图形之外，MATLAB 还允许绘制极坐标曲线、对数坐标曲线和直方图等，而这些曲线在某些场合是极为重要的。下面将分别介绍这些曲线的绘制。

(1) 函数 logplot 的用法同函数 plot 类似，绘制出的图形的横坐标、纵坐标均为对数坐标。

(2) 函数 semilogx 的用法同函数 plot 类似，绘制出的图形的横坐标为对数坐标，纵坐标为线性坐标。

(3) 函数 semilogy 的用法同函数 plot 类似，绘制出的图形的横坐标为线性坐标，纵坐标为对数坐标。

(4) 函数 polar(t，r，S)用于绘制极坐标曲线，其中 t 表示角度向量，r 表示幅值向量，S 表示选项，与 plot 函数类似。例如，键入下列命令：

```
t=0:.01:2*pi;
r=sin(2*t).*cos(2*t);
polar(t,r)
title('Polar Plot of sin(2t)cos(2t)')
```

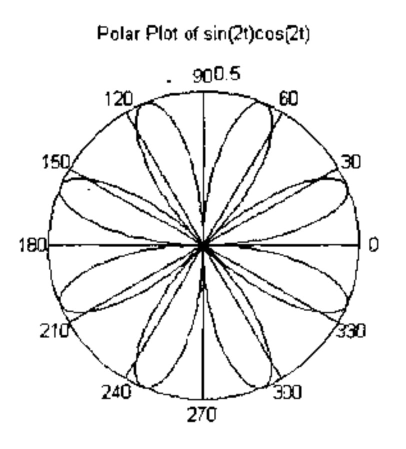


图 3-10 极坐标图形

则得到极坐标图形，如图3-10所示。

(5) 函数 bar( )用于绘制直方图；函数用于绘制梯形图。例如，在 MATLAB 的工作空间中键入下列命令：

```
x=-2.9:0.2:2.9;
y=exp(-x.*x);
bar(x,y)
```

则得到直方图如图3-11所示。

继续键入命令：

```
stairs(x,y)
```

则得到梯形图如图3-12所示。

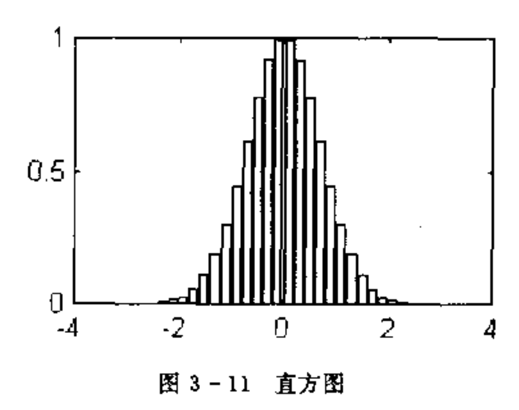


图 3-11 直方图

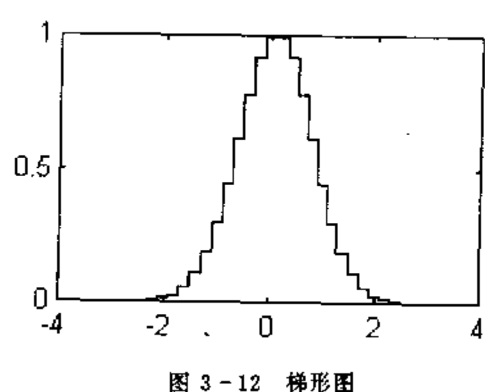


图 3-12 梯形图

3.8 三维图形的绘制

MATLAB 提供了丰富的函数来绘制三维图形，有的函数用来绘制三维曲线，有的函数用

来绘制三维曲面和轮廓图，此外还可以用伪色彩来表征第四维。

3.8.1 绘制三维曲线

和原来的二维图形相对应，MATLAB 提供了 plot3( )函数，可以在三维空间内绘制出三维的曲线，该函数的调用格式为

```
plot3(X1, Y1, Z1, S1, X2, Y2, Z2, S2,…)
```

其中，Xn、Yn、Zn 分别为维数相同的向量，分别存储曲线的三个坐标的值，而 Sn 选项可以用来定义曲线的颜色、线型、符号等信息，具体可参见表3－5。

下面举一个绘制三维曲线的例子。在 MATLAB 的工作空间中键入下列命令：

```
t=0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
```

由上面这段程序绘制出的三维曲线如图3－13所示。

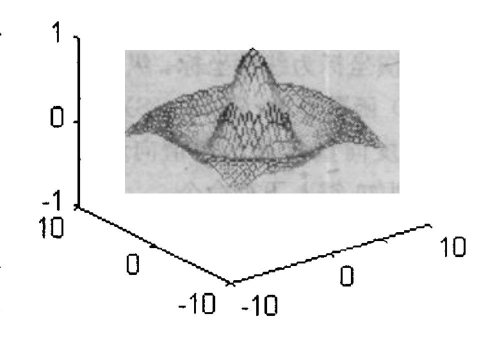


图 3－13 三维曲线图

3.8.2 三维曲面的绘制

mesh(Z)语句给出矩阵 Z 元素的三维消隐图，它所生成的网络形表面由生成 X—Y 平面的线格对应的 X 坐标所定义，图形由临接点用直线连接而成。mesh 用来显示以其它方式输出数据量太大的大型矩阵，也可绘制具有两个变量的图形函数。

为显示双变量函数 Z=f(x,y)的图形，第一步是产生特定的 X 和 Y 矩阵，它们在整个函数定义域中分别由重复的行和列构成，这可以利用函数 meshgrid( )来完成，例如：

```
[X,Y]=meshgrid(x,y)
```

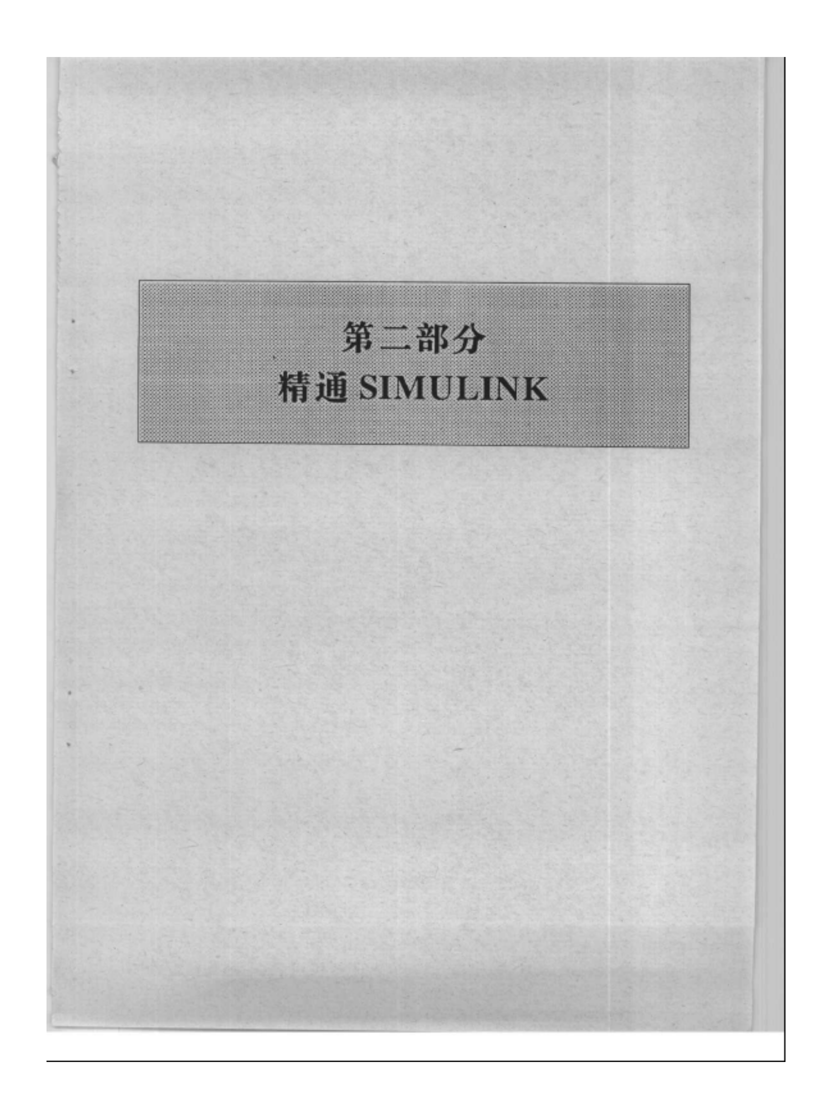
在这个基础上，函数便可直接计算和绘图。

考察 sin(r)/r 函数，它产生像“阔边帽”形状的图形，在 MATLAB 的工作空间中键入下列命令：

```
x=-7.5:0.5:7.5;
y=x;
[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(X,Y,Z)
```

图 3－14

第一条语句限定定义域和绘图的步长(由后面建立的 X、Y 矩阵的对应关系显示出来)，在其上计算函数值，第三条语句建立一个具有重复行的 X 矩阵以及产生相类似的 Y 矩阵，第四条语句产生矩阵 R，它含有矩阵中心即坐标原点至定义域中任一点(X,Y)的距离，之后计算 sin(R) / R 并用 mesh( )函数得出结果，如图3－14所示。



第二部分
精通 SIMULINK



第四章 SIMULINK 简介

4.1 什么是 SIMULINK

SIMULINK 是一个用来对动态系统进行建模、仿真和分析的软件包,它支持连续、离散及两者混合的线性和非线性系统,也支持具有多种采样速率的多速率系统。

SIMULINK 为用户提供了用方框图进行建模的图形接口,采用这种结构画模型就像你用笔和纸来画一样容易。它与传统的仿真软件包用微分方程和差分方程建模相比,具有更直观、方便、灵活的优点。SIMULINK 包含有 Sinks(输出方式)、Source(输入源)、Linear(线性环节)、Nonlinear(非线性环节)、Connections(连接与接口)和 Extra(其它环节)子模型库,而且每个子模型库中包含有相应的功能模块。用户也可以定制和创建用户自己的模块。

用 SIMULINK 创建的模型可以具有递阶结构,因此用户可以采用从上到下或从下到上的结构创建模型。用户可以从最高级开始观看模型,然后用鼠标双击其中的子系统模块,来查看其下一级的内容,以此类推,从而可以看到整个模型的细节,帮助用户理解模型的结构和各模块之间的相互关系。

在定义完一个模型以后,用户可以通过 SIMULINK 的菜单或 MATLAB 的命令窗口键入命令来对它进行仿真。菜单方式对于交互工作非常方便,而命令行方式对于运行一大类仿真非常有用(例如,用户要做蒙特卡洛仿真)。采用 Scope 模块和其它的画图模块,在仿真进行的同时,就可观看到仿真结果。除此之外,用户还可以在改变参数后来迅速观看系统中发生的变化情况。仿真的结果还可以存放到 MATLAB 的工作空间里做事后处理。

模型分析工具包括线性化和平衡点分析工具、MATLAB 的许多工具及 MATLAB 的应用工具箱。由于 MATLAB 和 SIMULINK 是集成在一起的,因此用户可以在这两种环境下对自己的模型进行仿真、分析和修改。

4.2 SIMULINK For Windows 的安装

4.2.1 系统要求

SIMULINK For Windows 依赖于 Windows 和 MATLAB 的要求,因而它所需要的系统配置和其它设置就不必专门增加或改变,只要能满足 Windows 系统的要求即可。具体来说,SIMULINK For Windows 的安装和使用对一些软硬件的要求如下:

(1) MATLAB 4.0 For Windows。

(2) 386及386以上的 PC 机。

(3) MS-DOS 3.1或更高版本。

(4) Microsoft Windows 3.1或更高的版本。

(5) 8 MB 的扩展内存。

(6) 硬盘有15 MB 的自由空间。

(7) 高密度的3吋软盘驱动器。

(8) Windows 支持的鼠标和监视器。

另外，如果系统中有以下几种设备或硬件，则性能更佳。

(1) 数字协处理器。

(2) 额外的存储器。

(3) 8位的图形适配器和显示器(256色)。

(4) Windows 支持的图形加速卡。

(5) Windows 支持的打印机。

(6) Windows 支持的声霸卡。

4.2.2 SIMULINK For Windows 的安装

在安装 SIMULINK For Windows 之前，要求系统必须先安装 Windows 和 MATLAB 4.0 For Windows，有关这两种软件的安装方法，请参阅有关的参考书。读者在 PC 机上已安装了这两种软件，然后按照下面的步骤安装 SIMULINK For Windows。

(1) 运行 Windows。

(2) 把装有 SIMULINK 的3吋高密度软盘插入磁盘驱动器中。

(3) 运行 Program Manager，并选择 File 菜单中的 Run 选项。

(4) 在 Command Line 提示行中，键入 A:setup(或 b:setup)然后用鼠标单击 OK 按钮。

(5) 在 Directory 提示行中，键入你所要安装 SIMULINK 的目录，然后用鼠标单击 OK 按钮。SIMULINK 默认的目录为 C:\MATLAB。

整个安装过程将持续几分钟，安装完毕后，你的目录中有以下几个子目录：

\SIMULINK　　SIMULINK 的 M 文件

\BLOCKS　　SIMULINK 的块和帮助文件

\SIMDEMOS　　SIMULINK 用于演示的 M 文件

第五章 SIMULINK 的快速入门

本章通过运行一个演示程序和创建一个简单的模型来帮助 SIMULINK 的新用户快速入门。

5.1 运行一个演示程序

与 SIMULINK 一起提供给用户的一个演示程序模拟的是一个房子的热动力特性。要运行这个程序，可采用下面的步骤：

(1) 用鼠标双击 MATLAB 图标来打开 MATLAB 程序。

(2) 在 MATLAB 的命令窗口键入 simulink 来运行 SIMULINK，这时 SIMULINK 就显示其所包含的子模型库，如图 5-1 所示。

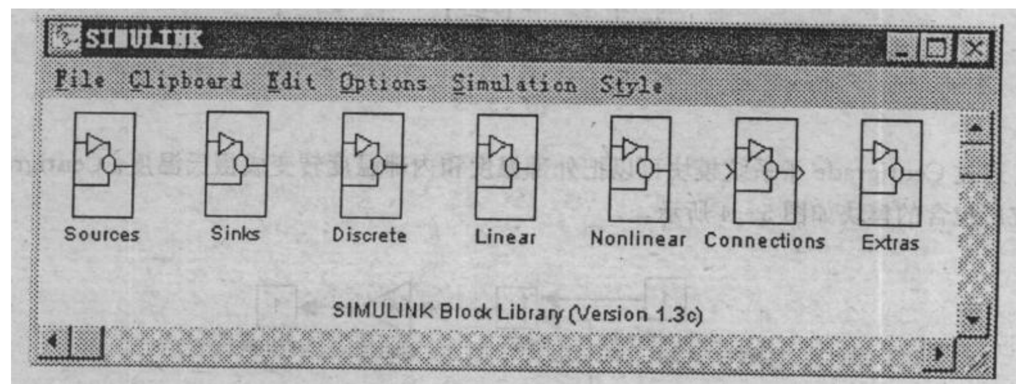


图 5-1

(3) 打开 Extras 子模型库来获得这个演示程序。方法是：在上面这个窗口的右边用鼠标双击Extras子模型库的图标；然后打开 Demos 子目录；用鼠标双击 mouse Thermostat 框。这时，SIMULINK 就创建一个新的窗口来显示这个模型，同时打开两个 Scope 模块。

(4) 用鼠标双击模型右上角的 Load Data 模块调入仿真需要的数据。

(5) 要进行仿真，在 Simulation 的下拉式菜单中选择 Start 命令，SIMULINK 就在 Indoor Temp Scope 中输出房子的温度，而在 Heat Cost Scope 模块中输出累积的热损耗。

(6) 要终止仿真可在 Simulation 下拉式菜单中选择 Stop 命令。

(7) 在 File 下拉式菜单中选择 Close 命令来关闭模型。

5.1.1 演示程序的说明

本演示程序采用一个简单的模型来模拟一个房子的热动力学特性。恒温器的温度设为 21℃，并且会受到外界温度干扰的影响，这个干扰用基本温度为 10℃、幅值为 -9℃的正弦波来描述。

1. 各子系统介绍

内部和外部的温度输入给 House 这个子系统，并不断地更新内部的温度。用鼠标双击

House 子系统块，就可看到这个子系统所包含的功能模块，如图 5－2 所示。

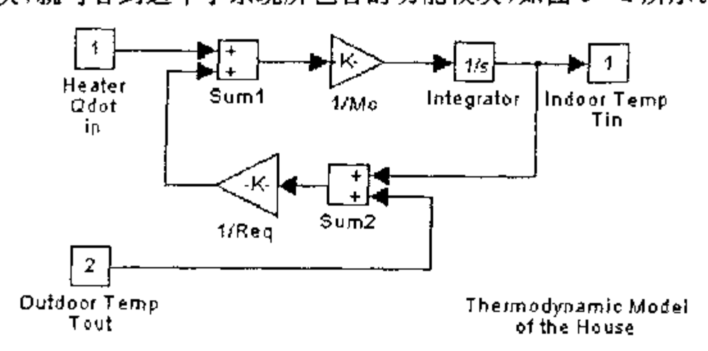


图 5－2 房子的热动力学特性模型

Thermostat 子系统用来模拟一个恒温器的工作，确定何时加热器打开或关闭，用鼠标双击 Thermostat 子系统模块，就可看到这个子系统所包含的模块，如图 5－3 所示。

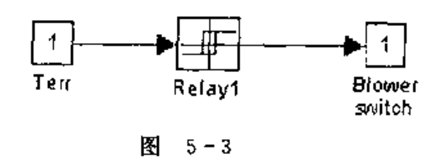


图 5－3

通过 Centigrade 子系统模块可以把外部温度和内部温度转变成摄氏温度。Centigrade 子系统所包含的模块如图 5－4 所示。

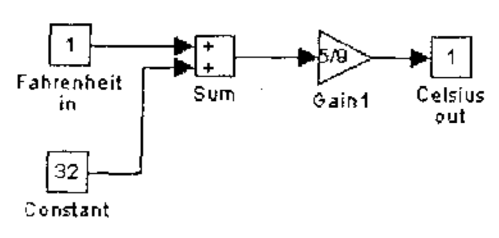


图 5－4

当加热系统工作的时侯，就计算其热损耗，并把所得的结果在 Heat Cost（$）Scope 模块上显示出来。同时，内部温度在 Indoor Temp Scope 模块上显示出来。

2. 仿真过程中模型参数的修改

在仿真过程中，也可修改模型中的参数来看系统的反应，例如：

（1）打开 Floating Scope 模块。Floating Scope 模块可允许用户选择模型中的任何连线来检查该连线中的信号。如果要查看它的工作方式，可在仿真开始以后，选择不同的连线来观察它所显示出来的信号。

（2）扩大 Scope 模块。Scope 模块包含有显示区域和用户用来控制显示信号的控制区域。水平轴代表时间，垂直轴代表信号。

（3）模型左上端标号为 Set Point 的模块是用来设置内部温度的希望值。在仿真进行的同时，把该块打开，并把其值重新设为 80℃，来观看内部温度和热损耗的变化情况。同样，也可改

变外部温度来看它对仿真的影响。

(4) 打开标号为 Daily Temp Variation 模块来调节日常温度，看改变幅值以后，仿真所发生的变化，等等。

3. 一些启示

从上面这个演示程序，可以得到以下一些有用的启示：

(1) 进行仿真需要包括两个步骤：一是定义参数，二是用 Start 命令启动仿真。定义仿真参数和进行仿真将在第七章中详细描述。

(2) 子系统(像 Thermostat、House、Temp Convert 等)可以把一组复杂的模块“隐藏”到一个子系统中。打开这些子系统模块，SIMULINK 就在一个新的窗口显示其所包含的基本模块。有关创建子系统的方法将在第六章中详细描述。

(3) Scope 模块就像示波器一样，用图形方式显示输出，它显示的是驱动块的输出。而悬浮的 Scope 模块，它与模型中的任何模块都不连，这样允许用户选择不同的连线来监视其中的信号，从而方便用户的使用。

(4) 模块中的参数可以由工作空间里的变量来定义。这些变量通过 Load Data 模块，采用 eval 命令调用 thermdat. m 文件，把数据放入工作空间里后，来获得它们的值。

5.2 创建一个简单的模型

如果不在 SIMULINK 环境中，可按照本章前面叙述的步骤来启动 SIMULINK。要创建一个新的模型，可在 File 菜单中选择 New 命令，SIMULINK 就创建一个新的窗口。用户可以将窗口移到屏幕的右上角，以便能同时看到窗口中的内容和子模型库中的内容。

在 SIMULINK 的窗口中用鼠标双击 Source 图标，这样就打开了 Source library，SIMULINK 就把 Source library 中所有的模块在一个新的窗口中显示出来，这些模块就是信号发生器。Source library 的窗口如图 5-5所示。

用户可以把库中或其它模型中的模块拷贝到自己的模型中。对于本节介绍的模型，需要拷贝的模块是 Source library 中 Signal Generator 模块。

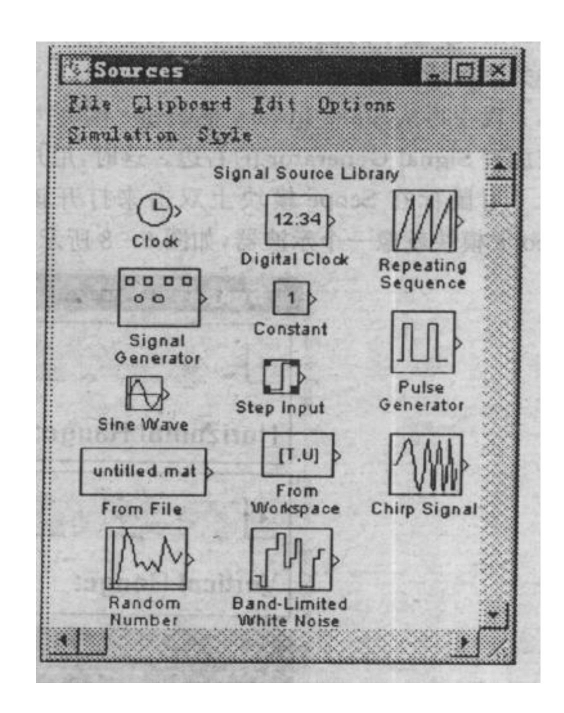


图 5-5

要从 Source library 中拷贝 Signal Generator 模块，可用鼠标单击这个模块，然后拖动鼠标把它移到自己的模型窗口中。当用户释放鼠标按钮以后，SIMULINK 就在用户的模型窗口中显示 Signal Generator 模块的图标。

用户可以用鼠标在模块上双击来查看模块中的参数和显示该模块的对话框。因此，在 Signal Generator 模块上双击后，就会看到如图 5-6 所示的对话框。

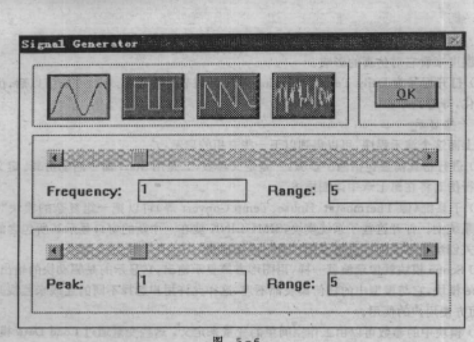


图 5-6

在对话框中用户可以选择由该模块产生的波形，并可定义其信号的幅值和频率。在把 Frequency 域中的值改为 6 以后，用鼠标单击 OK 按钮来接受该值，并关闭对话框。

现在再从 Sinks library 把 Scope 模块拷贝到自己的模型中，并把它放到 Signal Generator 的右边。这时，用户的模型如图 5-7 所示。

Signal Generator　　Scope

图 5-7

用鼠标在 Scope 模块上双击来打开该模块，这时你就会发现 Scope 模块就像一个示波器，如图 5-8 所示。



图 5-8

移动 Scope 窗口到一个合适的位置，然后把代表时间的 Horizontal Range 域中的值改为 10。下一步的工作是把两个模块连接起来。要把这两个模块连接起来，首先把鼠标指针定位到 Signal Generator 模块的输出端口并按下鼠标，然后拖动鼠标指针到 Scope 模块的输入端口并释放鼠标按钮。SIMULINK 就在两个模块之间画一条连线。如果连线不是一条直线，用户可以通过采用往上或往下移动模块的办法来使两者之间的连线变成直线，其中连线上的箭头代表信号的流向，如图 5-9 所示。

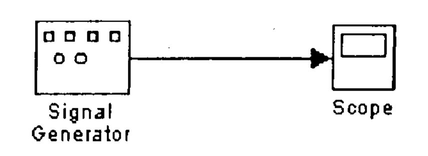


图 5-9

当用户建完模型并觉得满意以后，就可开始仿真。要选择仿真所采用的积分方法和参数，可以通过 Simulation 菜单中的 Parameters 选项来实现。SIMULINK 就显示 Control Panel 对话框，如图 5-10 所示。

SIMULINK Control Panel

Euler　Runge-Kutta 3　Runge-Kutta 5　Adams
Gear　Adams/Gear　Linsim

Start Time: 0.0
Stop Time: 999999
Min Step Size: 0.0001
Max Step Size: 10
Tolerance: 1e-3
Return Variables:

OK
Cancel

图 5-10

把参数 Maximum Step Size 改为 0.01，并接受其它的默认参数（默认的积分方法为 Runge-Kutta 5），然后用鼠标单击 OK 按钮来关闭对话框。这时就可通过 Simulation 菜单中的 Start 命令来启动仿真。

如果 Scope 模块的窗口没有打开，用鼠标在该模块上面双击后把它打开。那么 Signal Generator 模块在每步输出 sin(6t)后，就在 Scope 模块的窗口中显示该值。

要终止仿真有两种方式：一种是自动的方式，即仿真时间和 Control Panel 对话框中定义的时间相等时，就自动终止仿真；另一种是人工方式，即通过 Simulation 菜单中的 Stop 命令来终止仿真。

要退出 SIMULINK，有两种方法可以选择：一种是在 File 菜单中选择 Exit MATLAB 命令；一种是在 MATLAB 的命令窗口键入 Quit。

第六章　用 SIMULINK 创建模型

本章主要讨论用 SIMULINK 进行建模的方法和步骤，主要包括以下一些相关问题：

(1) SIMULINK 的窗口和菜单。

(2) 建模示例。

(3) 模型的创建。

(4) 模型的保存。

(5) 方框图的打印。

(6) 建模中的一些建议。

6.1　SIMULINK 的窗口和菜单

要运行 SIMULINK，用户必须首先运行 MATLAB。在 MATLAB 的提示符下，键入 simulink。用户的桌面上就显示一个包括有 MATLAB 命令窗口和 SIMULINK 的 BLOCK Library窗口的界面，其中在 SIMULINK 的 BLOCK Library 窗口中将显示各个库的图标，如图 6-1 所示。

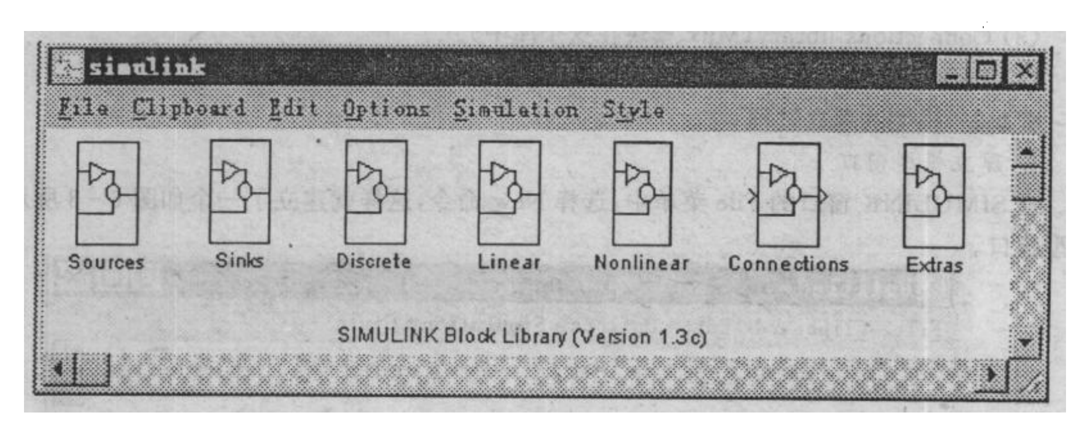


图　6-1

用户可以把库中的模块拷贝到模型窗口进行建模。所有的库及其有关的模块将在第九章中详细描述。

如果用户要进行仿真，并要分析仿真的结果，可在 MATLAB 的命令窗口键入 matlab 命令即可，有关这些方面的内容将在第七章中进行描述。

SIMULINK 具有各自的库、模型及仿真结果的图形输出窗口。SIMULINK 的窗口能根据常见的分辨率自动调整其大小。如果你的显示器具有很高的或很低的分辨率，你就会发现你的窗口要么很大，要么很小，用户可以用鼠标拖动边界来重新定义窗口的大小，然后在 File 菜单中用 Save 命令来保存新窗口的设置。

6.2 SIMULINK 建模示例

本例主要介绍如何用建模命令和如何操作来创建一个模型的问题。这个模型的功能是这样的：首先由 Signal Generator 模块产生一个正弦波，其中一路输出直接连到 Mux 模块的输入端，另一路输出通过 Gain 模块连到 Mux 模块的另一个输入端，从而形成一个二维的向量信号。这个向量信号一方面由 Scope 模块用来显示，另一方面保存到工作空间里的一个变量中。其具体框图如图 6－2 所示。

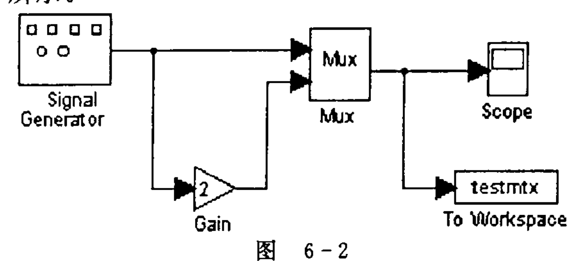


图 6－2

6.2.1 建模所需模块的查找

在这个模型中，用户可以从下面这些库中找到你所需的模块：

(1) Source library(Signal Generator 模块在这个库中)。

(2) Linear library(Gain 模块在这个库中)。

(3) Connections library(Mux 模块在这个库中)。

(4) Sinks library(Scope 模块和 To Workspace 模块在这个库中)。

6.2.2 模型的建立过程

1. 建立模型窗口

在 SIMULINK 窗口的 File 菜单中，选择 New 命令，这样就建立了一个如图 6－3 所示的模型窗口。

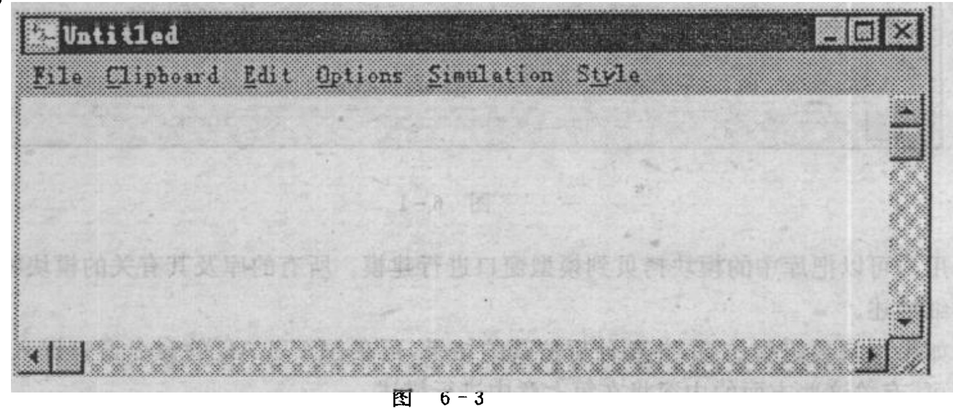


图 6－3

2. 向窗口拷贝模块

打开 Source library 库，来拷贝 Signal Generator 模块到模型窗口(图 6－4)。

要完成从一个库中拷贝一个块到模型窗口，需要以下几个步骤：

(1) 把鼠标移到所要拷贝的 Signal Generator 模块上面。

(2) 一直按下鼠标，这时你会发现鼠标形状就会发生改变，如图 6－5 所示。

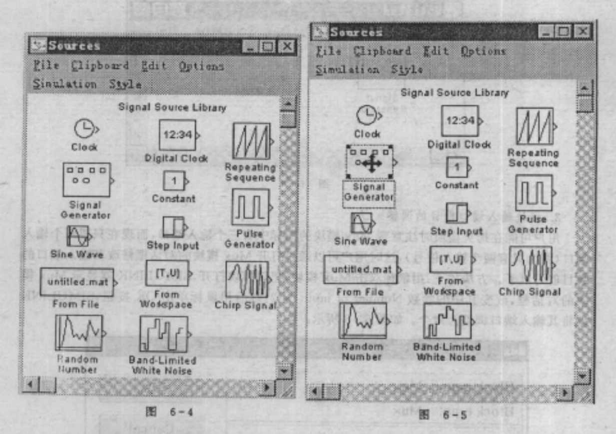


图 6-4　　　　　　图 6-5

(3) 一直按下鼠标并拖动 Signal Generator 模块到模型窗口(叫作 Untitled)，这时移动模块，你就会发现模块的轮廓和它的名字随着鼠标一起移动(图 6-6)。

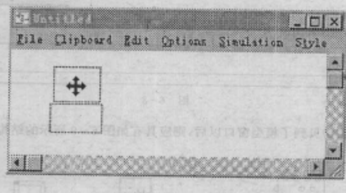


图 6-6

(4) 当鼠标指针已移到自己的模型窗口中，并在所需要放的位置时，松开鼠标。这时Signal Generator 模块就出现在自己的模型窗口中，如图 6-7 所示。

对于其它模块的拷贝和 Signal Generator 模块一样，在此不再赘述。用户也可以在模型窗口把一个模块从一个地方移到另一个地方，处理方法和前面拷贝模块时所用的鼠标拖动方法完全一样。

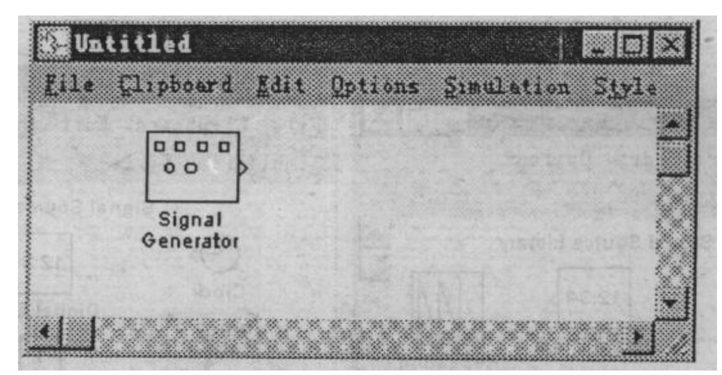


图 6-7

3. 模块输入端口数目的调整

用户可能在拷贝模块时注意到 Mux 模块的图标中有三个输入端口，而现在只要两个输入端口(因为只有两个输入信号)，这时用户可以通过打开 Mux 模块的对话框修改其输入端口的数目就可以了。方法如下：用鼠标双击 Mux 模块，把该模块打开，SIMULINK 就显示 Mux 模块的对话框，改变其中的参数 Number of input 为 2，然后用鼠标单击 OK 按钮，SIMULINK 就将其输入端口调整为两个。如图 6-8 所示。

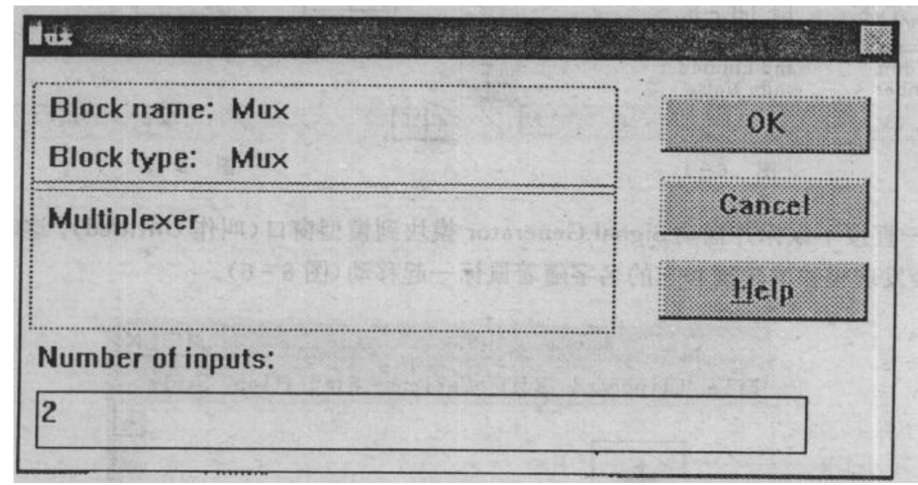


图 6-8

当所有的模块拷贝到了模型窗口以后，则应具有如图 6-9 所示的结构图。

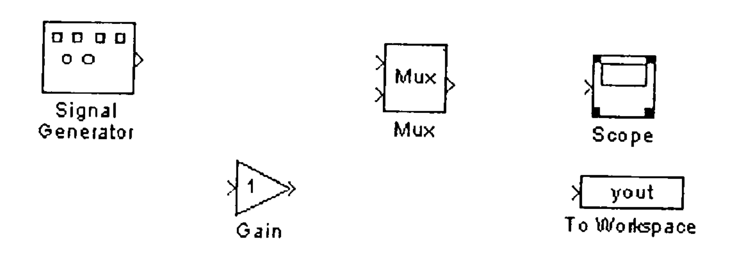


图 6-9

6.2.3 模块之间的连接线

剩下的任务是把它们连接起来。如果你仔细检查一下各个模块的图标，你就会发现在 Signal Generator 模块的右边有一个标记符号“>”，而在 Mux 模块的左边有两个“>”（图 6-10）。从块内部往外指的标记符号“>”代表模块的一个输出端口，从块外部指向块本身的标记符号“>”代表模块的一个输入端口。当模块被连接起来后，模块的端口符号“>”就自动消失。

下面具体介绍 Signal Generator 模块与 Mux 模块之间的连接和 Signal Generator 模块与 Gain 模块间的连接，其它模块之间的连接与这两模块之间的连接方法完全雷同。

首先介绍 Signal Generator 模块与 Mux 模块之间的连接，其步骤如下：

(1) 把鼠标指针移到 Signal Generator 模块右边的输出端口，如图 6-11 所示。

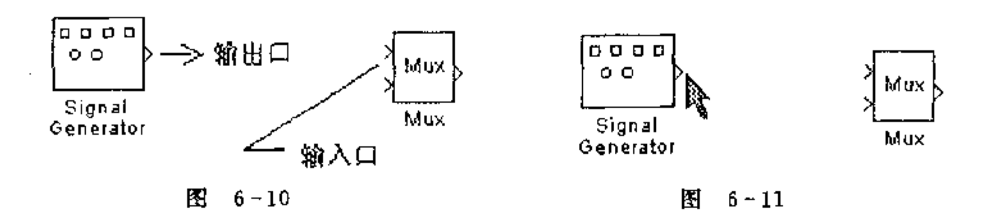


图 6-10　　　　图 6-11

(2) 按下鼠标按钮，这时鼠标指针变成十字型，如图 6-12 所示。

(3) 然后拖动鼠标到 Mux 模块左上边那个输入端口，这时鼠标指针仍保持为十字型，并有一条连线把 Signal Generator 模块和 Mux 模块上面的那个输入端口连接起来(图 6-13)。

(4) 释放鼠标按钮，这时两个模块就连接好了(图 6-14)。

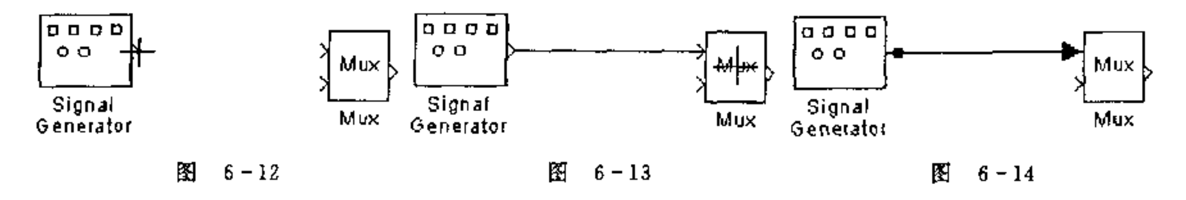


图 6-12　　　　图 6-13　　　　图 6-14

6.2.4 模块与其它两个模块间连接线的连接

回过头来，再看看本节开头的那个模型(图 6-2)，你就会发现大多数的连线是把模块的输出端口连接到其它模块的输入端口。然而，有两条连线是把“线”连到其它模块的输入端口，一条是把 Signal Generator 模块的输出端口连到 Gain 模块；另一条是把 Mux 模块的输出端口连接到 To Workspace 模块。这两条连线中流过的信号和它们各自的产生这些信号的“线”是一样的。换句话说，Signal Generator 模块给 Mux 模块和 Gain 模块相同的信号，Mux 模块给 Scope 模块和 To workspace 模块相同的信号。画这一类连线和前面画的连线稍微有所不同。下面就讲讲这类连线的画法。

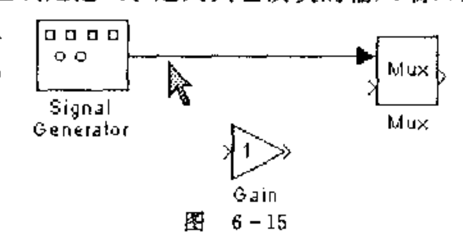


图 6-15

(1) 把鼠标放在 Signal Generator 模块与 Mux 模块之间的连线上，如图 6-15 所示。

(2) 在按住鼠标按钮的同时，按住 Ctrl 键，然后拖动鼠标到 Gain 模块的输入端口，如图 6-16 所示。

（3）释放鼠标按钮，SIMULINK 就从鼠标的起点位置到 Gain 模块的输入端口之间用一条线连接起来，如图 6－17 所示。

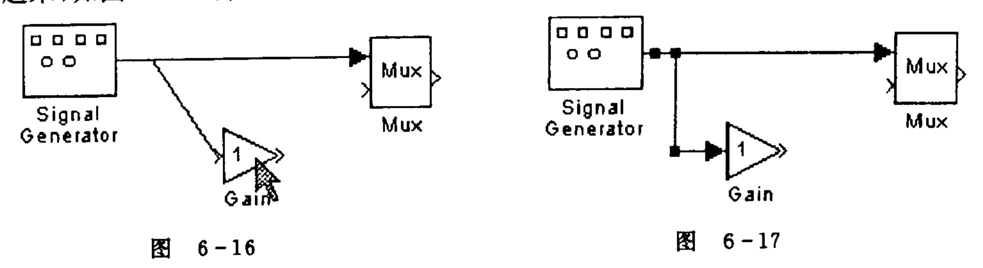


图 6－16　　　　图 6－17

6.2.5 模块对应参数的改变

当用户把所有模块之间的连线连接完毕以后，就可以改变各个模块（如果需要的话）所对应的参数，其步骤如下：

（1）打开 Gain 模块，把参数 Gain 设为 2，如图 6－18 所示。

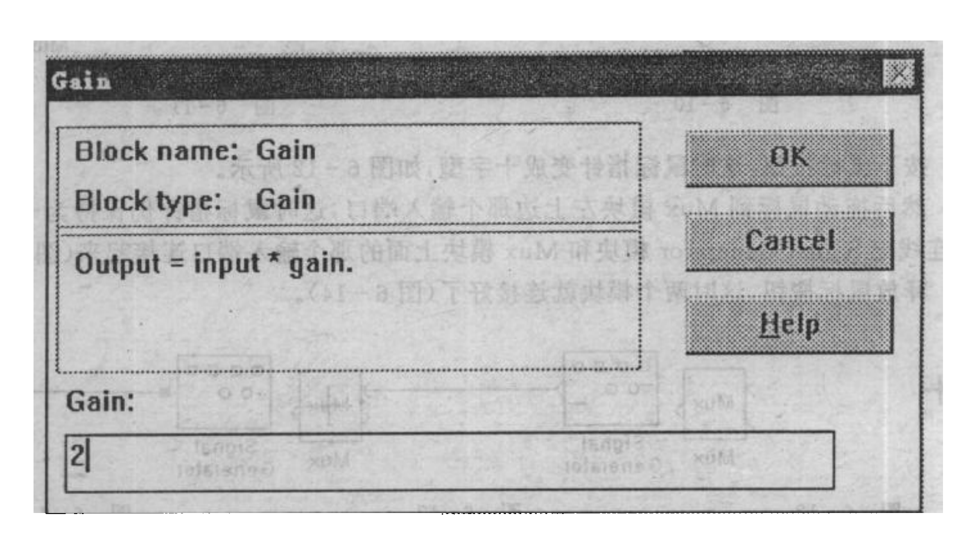


图 6－18

（2）打开 To Workspace 模块，把参数 Variable name 设为 testmtx（可以是其它的 MATLAB的变量名），这个变量用来保存仿真输出的结果。然后把参数 Maximum number of rows 设为需要的值，具体结果如图 6－19 所示。

（3）设置 Signal Generator 模块中的参数，方法同上。由于 Signal Generator 模块的默认输出幅值为 1 的正弦波，本例又只是一个示例，所以也就没有必要改变其值。因此，对该模块也就不做改变。

（4）在模型窗口的 Simulation 菜单中选择 Parameters 命令，于是出现如图 6－20 所示的控制面板的对话框。

分别选择积分方法为 Runge－Kutta 5、Start Time 为 0、Stop Time 为 10 s、Min Step 为 0.000 1、Max Step 为 0.1、Tolerence 为 1e－3、Return Variable 为空，然后用鼠标单击OK 按钮，控制面板对话框就如图 6－20 所示。

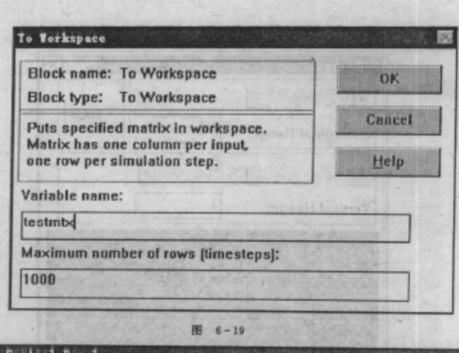


图 6-19

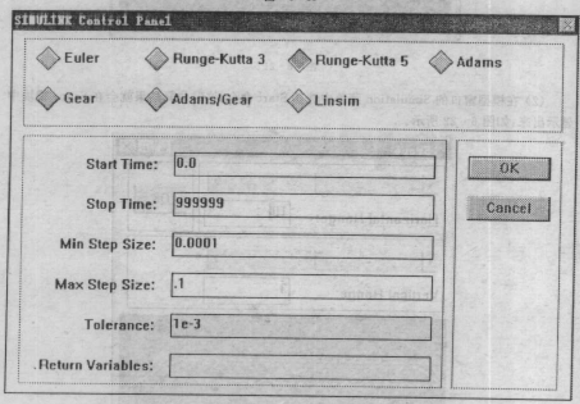
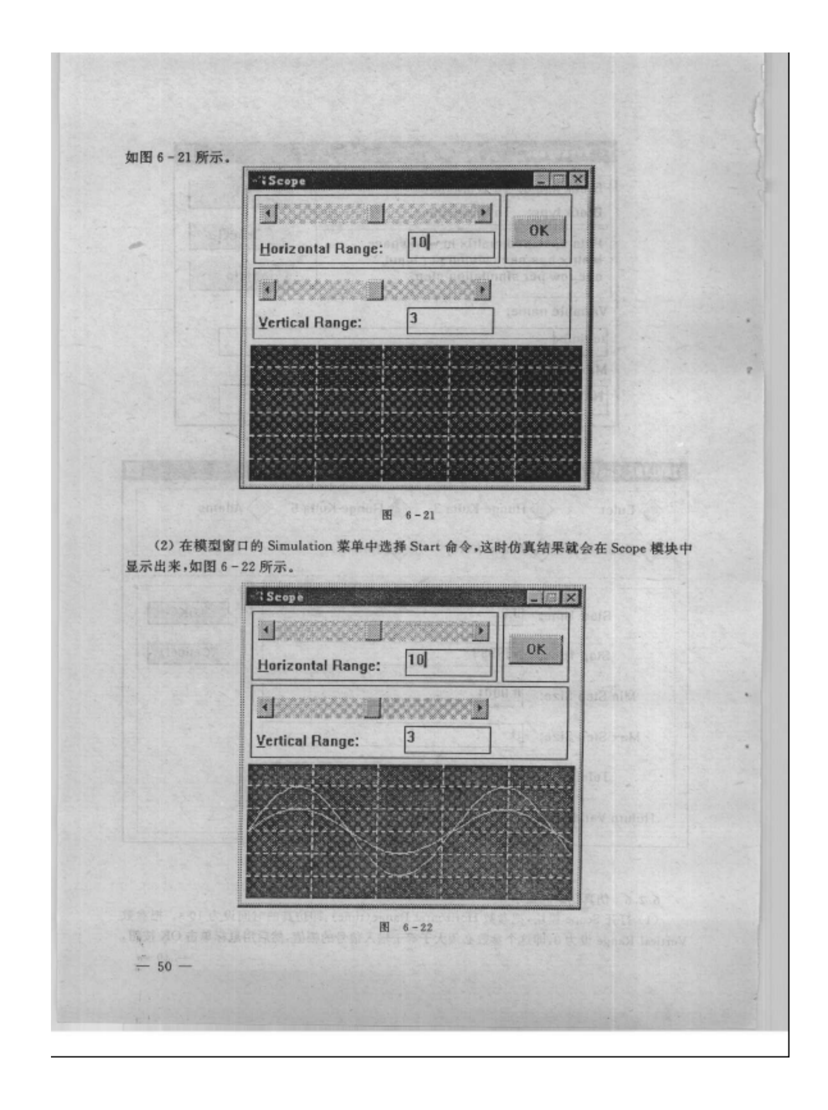


图 6-20

6.2.6 仿真结果的显示

(1) 打开 Scope 模块，把参数 Horizontal Range(time)，即仿真的时间设为 10 s。把参数 Vertical Range 设为 3，即这个参数必须大于等于输入信号的幅值，然后用鼠标单击 OK 按钮，



如图 6-21 所示。

图 6-21

(2) 在模型窗口的 Simulation 菜单中选择 Start 命令，这时仿真结果就会在 Scope 模块中显示出来，如图 6-22 所示。

图 6-22

6.2.7 模型保存与结束运行

如果需要把这个模型保存起来，可以在模型窗口的 File 菜单中选择 Save 命令，然后输入要保存这个模型的文件名（其扩展名为 M），用鼠标单击 OK 按钮后就可把这个模型保存起来。

如果要结束运行 SIMULINK 和 MATLAB，可以选择 File 菜单中的 Exit Matlab 命令即可。

作为一个示例，本例只涉及到最基本的一些方法用于建模，有关详细的细节，将在本章余后的章节中叙述。

6.3 建立模型

本节将详细论述建立用户自己的模型所要完成的任务。如果用户要建立一个新的模型，你可以像其它 Windows 程序一样移动、改变窗口的大小。如果要编辑一个已经存在的模型框图，则有下面两种方法供你选择：

(1) 在 SIMULINK 的 File 菜单中选择 Open 命令，然后键入存放模型的文件名（其扩展名为 M）。

(2) 在 MATLAB 的命令窗口键入存放模型的文件名。

这样，SIMULINK 就会为用户建立一个新的模型窗口，并把模型在这个窗口中显示出来。

6.3.1 选择对象

在建模和编辑过程中，经常遇到选择一个或多个模块和连线（这些模块和连线统称为对象）的问题。下面就具体谈一谈选择对象的方法。

一、选择一个对象

要选择一个对象，只要用鼠标在对象上面单击一下即可。这时你就会发现被选中对象的角上有一个“把手”标记。例如，图 6-23 表示 Sine Wave 块和一条连线被选中。

图 6-23

当用户在某个对象上面用鼠标单击后选中了这个对象，那么任何以前已被选中的对象就不再被选中。

二、选择一个以上对象

用户可以选择一个以上对象，这有两种方法可以选择：一种方法是逐个选择法；另一种方法是用方框选择相邻的几个对象。下面分别介绍这两种方法。

1. 逐个选择法

要选择一个以上对象，只要按下 Shift 键，然后在要选择的对象上面用鼠标逐个单击一下，那么凡是用鼠标单击过的对象就都被选中；反之，如果你要放弃某个已被选中的对象，那么只需在按下 Shift 键的同时，在已被选中的对象上面用鼠标单击一下，那么这个对象就不再被选中。

2. 用方框选择一个以上对象

选择一个以上的对象，一种比较简单的方法是在要选择的对象周围用方框框起来。要定义一个这样的方框，可采用下面的步骤：

(1) 首先定义方框的起始角，方法是把鼠标指针移到方框的一个起始角上，然后按下鼠标按钮，如图 6-24 所示。

(2) 拖动鼠标,把鼠标指针移到方框起始角的对角上,如图 6-25 所示。

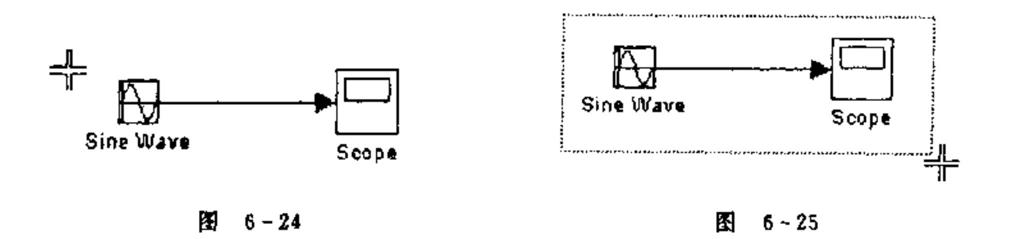


图 6-24　　　　图 6-25

(3) 释放鼠标按钮,那么在方框里面的所有模块和连线就都被选中,如图 6-26 所示。

3. 选择整个模型

要想在模型的活动窗口中选择所有的对象,只要在模型窗口的 Edit 菜单中选择 Select All 命令即可。

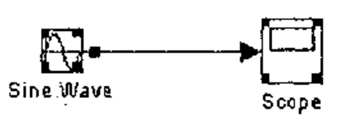


图 6-26

6.3.2 模块的操作

本节讨论建模中有关模块的操作问题,下面对这些内容分别予以介绍。

一、两个窗口之间的模块拷贝和移动

1. 用鼠标拷贝模块

当用户建立自己的模型时,经常遇到这样的问题,即从 SIMULINK 库中或其它的模型窗口中把模块拷贝到自己的模型窗口中。要解决这样的问题,可以采用下面的步骤:

(1) 打开相应的库或源模型窗口。

(2) 把用户所要拷贝的模块拖到自己的目标模型窗口中,方法是先用鼠标选中你所要拷贝的对象,然后拖动鼠标,并把鼠标指针移到自己的模型窗口中,然后释放鼠标按钮即可。有关如何选择对象的问题请参见 6.3.1 的内容。

2. 用菜单中的命令拷贝模块

用户也可以用 Edit 菜单中的 Copy 和 Paste 命令来拷贝模块,其方法如下:

(1) 选择你所要拷贝的块。

(2) 在 Edit 菜单中选择 Copy 命令。

(3) 选择你的模型窗口,使之成为活动窗口。

(4) 选择 Edit 菜单中的 Paste 命令。

这样你所要拷贝的模块就出现在自己的模型窗口中。

3. 模块的命名

SIMULINK 会给每一个拷贝过来的模块进行命名。如果在整个模型当中用户所拷贝过来的模块是这一类型模块中的第一块,那么它的名字和原来所在窗口中的名字是完全相同的。例如,如果用户把 Gain 模块从 Linear 库中拷贝到自己的模型窗口中,那么这一模块的名字就是 Gain。如果用户的模型中已经包含了一个 Gain 模块,那么 SIMULINK 就在模块名字的后面跟上一个按照顺序排列的数字(如 Gain1、Gain2,等等)。用户也可以对模块进行重新命名,这将在后面论述。

4. 拷贝模块之间的连线

如果用户要拷贝一个模块,而这个模块又和其它模块连接着,那么 SIMULINK 不会拷贝

它们之间的连线。如果用户要拷贝与这个模块相连的那个模块，那么它们之间的连线也将一起拷贝过来。

当用户拷贝一个模块时，那么拷贝过来的模块将继承源模块中的所有参数值。

5. 模块的移动

SIMULINK 采用一个不可见的 5 个像素的网格来简化对象的移动。模型中的所有模块和网格中的一条线对齐。用户也可以采用上、下、左、右 4 个箭头键来慢速移动一个模块的位置。

用户也可以用 Copy、Cut、Paste 命令来拷贝或移动模块，这时只是它们的图形被拷贝过来，而其参数不进行拷贝。

要在一个窗口把一个以上的模块拷贝到另一个窗口，其方法和拷贝一个模块是类似的，唯一的区别是在你选择模块时，按下 Shift 键。

二、在同一个模型中移动模块

在同一个模型窗口中把一个模块从一个地方移到另一个地方，其方法很简单。首先是选中该模块，然后拖动鼠标指针到一个用户所希望的位置，最后释放鼠标按钮即可。SIMULINK 会自动重新布置连到该模块上的连线。

要移动一个以上的模块(包括它们之间的连线)，其方法如下：

(1) 选中用户所需要移动的模块和连线，其方法参见上一节的内容。

如果用户用逐个选择法来选中一个以上的对象，在你选中最后一个对象后，不要释放鼠标按钮。否则的话，当你用鼠标单击一个已被选中的对象时，将导致那个对象不在选中之列。

如果用户用定义方框的办法来选中一个以上的对象，用户只需用鼠标单击任何一个你已选中的模块，那么在你拖动该模块的同时，就等于拖动方框内的所有对象。注意不要用鼠标单击方框内的连线，否则的话，只有那条连线被选中。另外，如果你在模型窗口中任何对象未曾占有的区域内单击一下，那么所有的对象就不再被选中。

(2) 拖动选中的模块和连线到你所希望的位置，然后释放鼠标按钮。如果你把一个模块移到了一条连线上面，你可以重新布置连线，方法是：首先选中该连线，然后在 Option 菜单中选择 Reroute Lines 命令即可。SIMULINK 会重新布置这根连线，使之从模块旁边通过。

三、在同一个模型中复制模块

在同一个模型中复制模块的方法如下：

(1) 按下 Ctrl 键，然后用鼠标左键把要复制的模块选中。

(2) 用鼠标把要复制的模块移到你所要复制的位置。

这时就在要复制的位置上把原来的模块复制了一份。

四、定义模块中的参数

一个模块的某些功能是由模块中的参数来定义的，用户可以通过其对话框来定义这些参数值。用户只要用鼠标双击该模块把它打开，SIMULINK 就显示该模块的对话框，并列出这些参数和它们的当前值。用户可以改变这些值或接受这些默认值。

有关各个模块的对话框及其参数的含义将在第八章中详细讨论。

五、删除模块

要删除一个或几个模块，首先必须选中需要删除的模块，然后按 Delete 键或者从 Edit 菜单中选择 Clear 或 Cut 命令。Cut 命令把要删除的模块移到了剪辑板上，而 Delete 键或 Clear 命令则不会影响剪辑板上的内容。

六、断开模块间的连接

要在一个模型中断开与一个模块之间的连接，而并不删除它，方法是按下 Shift 键，然后选中该模块，并从模型中的原始位置拖动该模块即可。

七、改变模块的方向

SIMULINK 默认，信号从模块的左边流进，从模块的右边流出，即输入在左边，输出在右边。用户也可以采用下面的办法来改变模块的方向：

(1) 采用 Option 菜单中的 Rotate 命令，每运行一次，就顺时针方向转动 90 度。

(2) 采用 Option 菜单中的 Flip Horizontal 命令，每运行一次，就转动 180 度。

(3) 采用 Style 菜单中的 Orientation 命令，用户可以选择模块的方向，包括 Left to Right、Right to Left、Up 或 Down。

图 6-27 用来演示 SIMULINK 采用 Rotate 和 Flip 命令来改变模块的方向时各个口的变化情况。

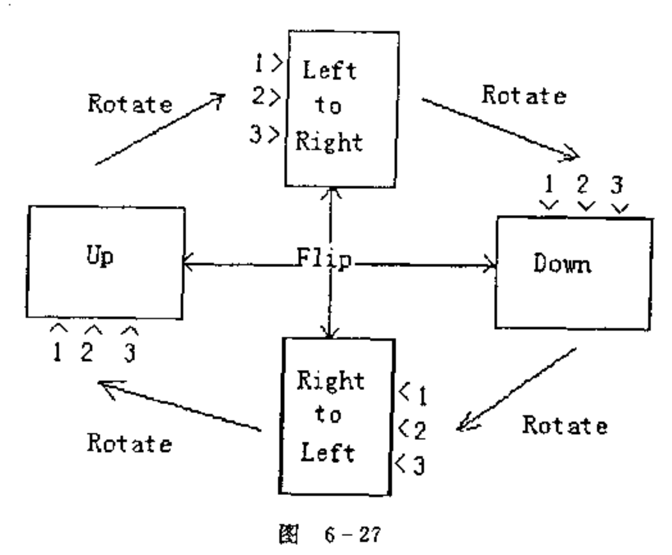


图 6-27

八、重新定义模块的大小

要改变一个模块的大小，首先必须选中它，然后用鼠标拖动该模块的任何一个手柄即可。一个模块最小可定义为一个 5×5 的像素，而最大只受计算机屏幕的限制。当用户改变一个模块大小的时候，一个形状如箭头的标志表示你所拖动的角和你所拖动的方向。当模块被重新定义大小的时候，一个长方形虚框表示其真正的大小。

例如，图 6-28 表示重新定义 Signal Generator 模块的大小。其含义是选中了右下角的手柄，且朝箭头的方向进行拖动。当鼠标按钮释放以后，长方形虚框的大小就是 Signal Generator 模块真正的大小。

Signal Generator

图 6-28

九、模块名的操作

在一个模型当中，所有的模块名必须是唯一的，而且必须至少含有一个字符。SIMULINK 默认，如果一个模块的端口在其右边的话，那么它的名字就在它的下面；如果一个模块的端口在其上边或下边的话，那么它的名字就在它的右边，如图 6-29 所示。用户也可以改变模块名和模块名的位置。

1. 改变模块名

用户可以采用下面三种方法之中的任何一种来改变模块的名字：

(1) 选中显示名字的盒子，然后键入新名字。

(2) 把插入指针移到名字中间，然后插入新的字符。

(3) 拖动鼠标，选中所要替换字符所在的区域，然后键入新字符。

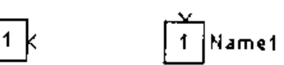


图 6-29

当用户用鼠标在其它模块上单击或者进行其它的操作后，改变后的名字要么得到承认，要么被拒绝接受。如果用户试图改变一个模块名为一个已经被其它块占有的名字，或者是一个没有字符的名字，SIMULINK 就显示一个出错信息。

用户可以修改一个模块名或几个模块名的字体，方法是：首先选中这些模块，然后在 Style 菜单中选择 Fonts 子菜单，从这个子菜单中选择你所需要的字体即可。

2. 改变模块名的位置

用户可以改变一个模块名的位置，不管这个模块名是显示的还是隐藏的，方法是在 Style 主菜单中选择 Title 子菜单。在这个子菜单中有以下几个选项：

(1) Displayed。SIMULINK 默认为显示模块名。

(2) Hidden。不显示模块名。

(3) Top/Left。如果一个模块的方向为 Left to Right 或者是 Right to Left，则把模块名放在模块的上边；如果一个模块的方向是 Up 或者是 Down，则把名字放在模块的左边。

(4) Bottom/Right。如果一个模块的方向为 Left to Right 或者是 Right to Left，则把模块名放在模块的下边；如果一个模块的方向是 Up 或者是 Down，则把名字放在模块的右边。

图 6-30 演示的是 Top/Left 模块名的位置。

有关模块的方向问题，请参见 6.3.2 之七“改变模块的方向”这一小段。

图 6-30

十、模块的向量化

几乎所有的模块既能接受标量输入，也能接受向量输入，并允许用户来定义标量或向量参数。在第九章中将详细讨论模块的输入、输出及参数的特性。

用户可以通过 Style 菜单中的 Wide Vector Lines 选项来定义模型中的哪一条线传递的是向量信号，SIMULINK 就把传递向量信号的线比传递标量信号的线画得宽一些。下一节中的那张图指出了宽线和细线的区别。当用户选择了上面的选项，并希望更新模型的显示，用户就必须在 Style 菜单中选择 Update Diagram 选项来更新显示。另外，在仿真开始时也进行这样的更新显示。

十一、输入和参数的标量扩展

标量扩展是指把一个标量变成一个具有相同元素的向量。SIMULINK 只对一个模块的输入和参数进行标量扩展。第九章中在对每个模块的说明中，都指出了 SIMULINK 是否对该模块的输入、参数或者两者同时进行标量扩展。

1. 输入的标量扩展

当某个模块(如 Sum、Relational Operator 等)有一个以上的输入时，用户可以把向量输入和标量输入混合起来。在这种情况下，那个标量输入信号就要进行标量扩展，形成一个具有和向量输入信号维数一样的具有相同元素的向量。下面这个例子用来演示 Sum 模块的一个输入

进行标量扩展后形成一个向量输入的情况，请注意较粗的那些向量线(图 6－31)。

2. 参数的标量扩展

对于可以进行标量扩展的那些模块，其参数既可以定义为标量，也可定义为向量。当定义为一个向量参数时，向量参数中的每一个元素与输入向量中的每一个元素相对应。而当定义为一个标量参数时，SIMULINK 就对标量参数进行标量扩展，自动形成一个具有相应维数的向量。

下面这个例子用来演示 Gain 模块的一个标量参数进行标量扩展后，形成一个与输入向量维数相同的具有 3 个元素的向量参数的情况(图 6－32)。

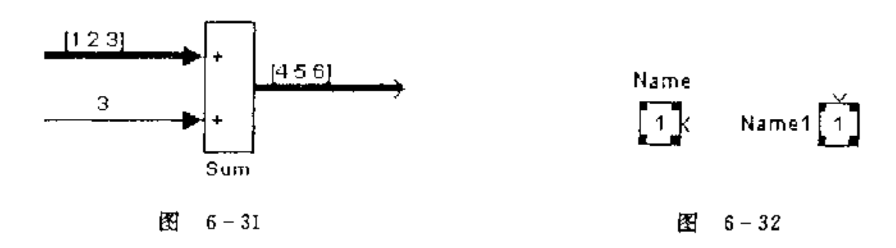


图 6－31　　图 6－32

6.3.3 连线的操作

连线既可以把一个模块的输出和另一个模块的输入连接起来，也可以把其它的连线与一个模块的输入连接起来。每一个输出端口可以有任意条连线，而每一个输入端口却只能有一条连线(Mux 模块可把几个标量连线合并成一个向量连线，这个功能非常有用，有关其详细的描述请参见第九章)。

一、块间连线

要把一个模块的输出和另一个模块的输入连接起来，可采用下面的步骤：

(1) 把鼠标指针移到第一个模块(如下图的 Constant)的输出端口。其实用户只需把鼠标指针移到靠近端口的任何位置，而没有必要把指针非常精确地移到那个端口位置上，如图 6－33 所示。

(2) 按下鼠标按钮，这时鼠标指针就变成十字形，如图 6－34 所示。

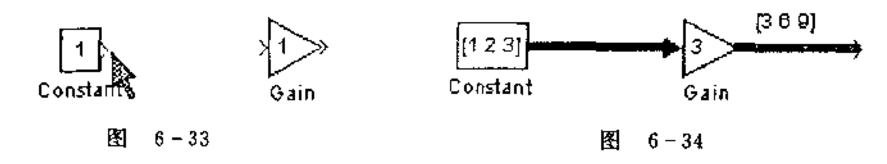


图 6－33　　图 6－34

(3) 拖动鼠标指针到第二个模块(Gain)的输入端口。用户只需把鼠标指针移到靠近模块输入端口的任何位置。如果用户把鼠标指针移到了第二个模块的里边，则把连线连到第二个模块第一个未曾占有的输入端口。如果想把连线连到指定的端口，就必须在释放鼠标按钮以前把鼠标定位到那个输入端口(图 6－35)。

(4) 释放鼠标按钮。SIMULINK 就用一个带箭头的连线代替端口的符号，用来表示信号的流向。用户既可以把连线从一个模块的输出端口连到另一个模块的输入端口，也可以反方向连接。在这两种情况下，箭头都出现在输入端口，而且信号的流向不会因为连线的方向不同而发生改变(图 6－36)。



图 6-35　　　　图 6-36

1. 在模块的周围布线

SIMULINK 在布线时自动在模块的周围进行布线，而不是把连线从模块的中间穿过。不过，如果用户在画连线或线段的同时，按下了 Shift 键，那么 SIMULINK 就根据用户的要求进行连线，不再遵循上面的原则。如果用户的框图中有互相交叠的区域，而用户又想把它清除掉，可以通过选择 Reroute Lines 选项来实现。具体方法是用定义方框的办法选中这个区域，然后在 Options 菜单中选择 Reroute Lines 命令即可。

2. 从其它连线画连线

用户可以在一条已经存在的连线上引出另一条连线，那么这两条连线就传送相同的信号给它们各自的对象。

例如，图 6-37 所示的两张框图中，左边的那张框图有一根单线，它把 Product 模块的输出端口连到了 Scope 模块的输入端口。而右边的那张框图增加了一根连线，这根连线把 Product 模块的输出端口连到了 To Workspace 模块的输入端口。事实上，Product 模块和 Scope 模块之间的连线与 Product 模块和 To Workspace 模块之间的连线上所流过的信号都是 Product 模块的输出信号。

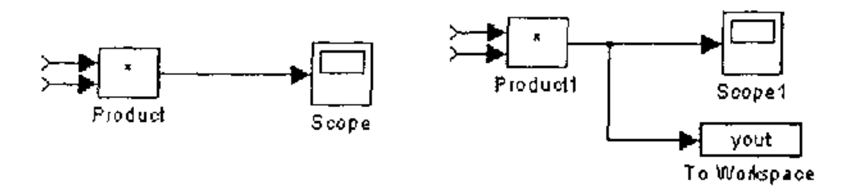


图 6-37

要从某一根连线上引出另一根连线，可采用下面的方法：

(1) 把鼠标指针移到这根连线上的某个位置，这个位置就是引出新线的起始位置。

(2) 在按下 Ctrl 键的同时，按下鼠标按钮。

(3) 拖动鼠标到目标端口，然后释放鼠标按钮和 Ctrl 键，那么 SIMULINK 就在起始位置和目标端口之间创建了一条新线。

用户也可在按下鼠标左键的同时，按下鼠标右键，替代按下 Ctrl 键的功能来引出新线。

3. 画一根线段

画一根线段，就等价于画一根连线，而这根连线不连到框图中的任何一个对象上。在画完一根线段后，就在线段的末端出现一个箭头，以表示该线段还没有终止。如果要在刚才的线段后面接着画线段，只要把鼠标移到箭头上面并按下鼠标，其余的就是重复上面的步骤。如果模型中有任何未曾连完的线段，仿真时 SIMULINK 就给出警告信息。

图 6-38 就表示框图中有一根没有连完的线段。

4. 连线的角度

除了下面这些情况以外，SIMULINK 总是以 45 度的倍数的角度来画连线。

(1) 如果鼠标指针移到了一个可用端口的附近，那么那根连线就连到该端口。

(2) 如果在创建新线的同时，按下了 Shift 键，SIMULINK 就根据用户的要求进行画线。

图 6-38

(3) 如果用户是通过移动鼠标指针到某个模块的上面并且释放鼠标来创建新线的话，那么这根连线就连到该模块最上面或最左边未曾占用的端口上。

二、删除连线

要删除一根或几根连线，首先是选中它们，然后按下 Delete 键或者是在 Edit 菜单中选择 Clear 或 Cut 命令即可。

三、移动线段

要移动一根线段，可采用下面的步骤：

(1) 把鼠标指针移到线段上面，如图 6-39 所示。

(2) 按下鼠标按钮，这时鼠标指针变成十字形，如图 6-40 所示。

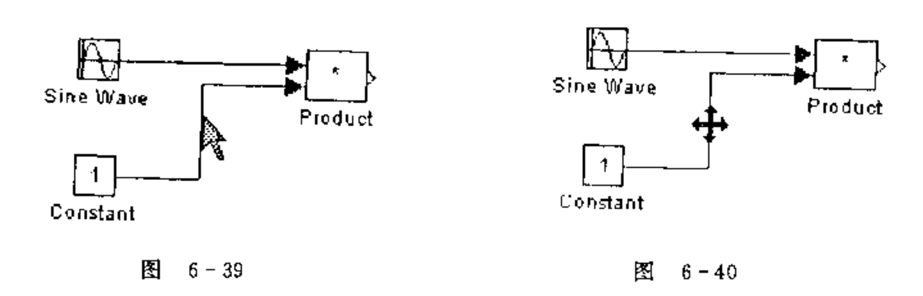


图 6-39　　图 6-40

(3) 拖动鼠标指到用户所希望的位置(图 6-41)。

(4) 释放鼠标按钮(图 6-42)。

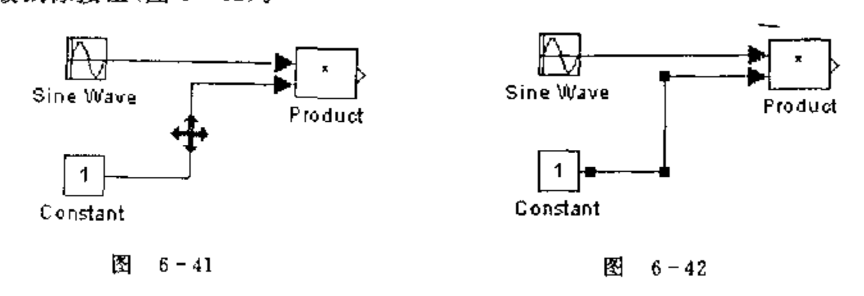


图 6-41　　图 6-42

注意：用户不能移动直接连到某个模块端口上的线段。

四、移动顶点

要移动一根连线的顶点，首先是把鼠标指针移到该顶点上面，然后按下鼠标按钮，并拖动鼠标指针到用户所希望的位置，最后释放鼠标按钮即可。

注意：用户不能移动一根连线上两个端点处的这些顶点。

图 6-43 用来演示拖动鼠标时鼠标的形状和顶点的移动情况，用户可以向任何方向移动顶点。

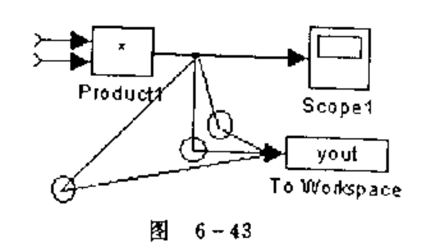


图 6-43

五、把一根连线分割成线段

用户可以把一根连线(或者一根线段)分成两根线段,而把这根连线的两个端点仍放在原来的位置。SIMULINK 就会自动创建这些线段和连接这些线段的顶点。要把一根连线分割成线段,可采用下面的步骤:

(1) 把鼠标指针移到连线上的某一点,且这一点是你所需要的顶点,如图 6-44 所示。

(2) 按下 Shift 键的同时,按下鼠标按钮(图 6-45)。

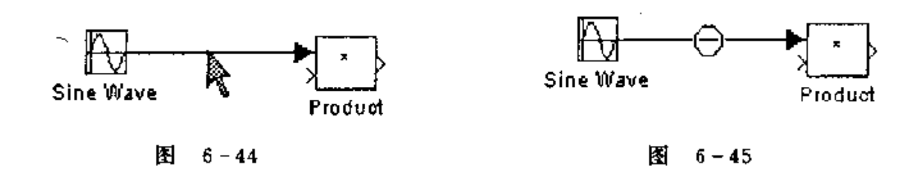


图 6-44　　　　图 6-45

(3) 拖动鼠标指针到用户所希望的位置(图 6-46)。

(4) 释放鼠标按钮和 Shift 键(图 6-47)。

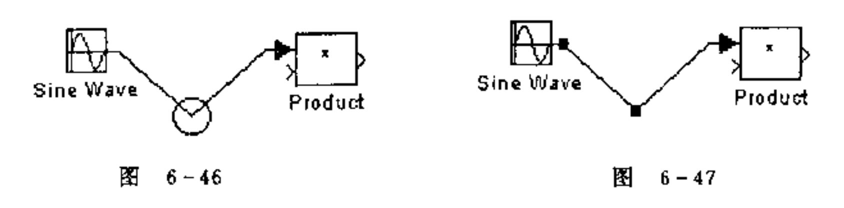


图 6-46　　　　图 6-47

6.3.4 鼠标和键盘操作总结(表 6-1)

表 6-1

建模操作	鼠标和键盘操作
选择对象	鼠标左键
选择一个以上对象	Shift+鼠标左键
从另一个窗口拷贝对象	选中对象并拖动鼠标指针
移动对象	选中对象并拖动鼠标指针
复制对象	Ctrl+鼠标左键,然后拖动鼠标或鼠标右键,然后拖动鼠标
模块间连线	鼠标左键
断开模块间连线	Shift+拖动模块

续 表

建模操作	鼠标和键盘操作
在模块周围布线	Shift＋画线
从其它线引线	Ctrl＋拖动连线
移动线段	选中线段并拖动鼠标指针
移动顶点	选中顶点并拖动鼠标指针
把连线分割成线段	Shift＋拖动线段

6.3.5 给模型框图添加文本注释

一、添加文本注释

用户可以在模型的框图中添加文本注释，方法是：

(1) 把鼠标指针移到需要添加文本注释的位置。

(2) 单击一下鼠标按钮。

(3) 键入注释文本。

(4) 再单击一下鼠标按钮。

SIMULINK 就把文本注释放在用户单击鼠标时鼠标指针所在位置略微偏下一点的地方。

注意：文本注释在模型中必须是唯一的。

二、修改文本注释

用户也可以修改文本注释的字体，方法是：

(1) 用定义方框的办法来选中你需要修改的文本注释。

(2) 在 Style 菜单中选择 Fonts 子菜单。

(3) 在 Fonts 子菜单中选择你所需要的字体。

6.3.6 创建子系统

一、模块的分组

随着系统规模和复杂性的增加，模型也在不断增大。为了使问题得到简化，可以对各个模块进行分组，然后把各组分别组成一个子系统的办法来使复杂问题简单化。对各个模块进行分组的好处主要有以下一些原因：

(1) 有助于在模型窗口中减少模块的个数。

(2) 有助于把功能有关的模块编入同一组。

(3) 有助于建立递阶结构的框图。

用户可以有两种方法来建立子系统：① 在模型窗口中添加一个 Subsystem 块，然后把该模块包含的模块添加进去即可；② 如果组成一个子系统的模块已经添加进去了，那么可把这些模块归入到同一个子系统中。

二、模块的创建

如果在一个子系统所包含的模块中还未加入以前要创建的子系统，用户首先必须添加一个 Subsystem 模块，然后再创建组成子系统的各个模块，具体方法如下：

(1) 从 Connections 库中把 Subsystem 模块拷贝到用户模型窗口中。

(2) 双击 Subsystem 模块，使之打开。

(3) 在空白的子系统窗口，创建子系统。用 Inport 模块表示从子系统外面输入，用 Output 模块表示子系统的输出。例如，下面框图中的 Sum 模块是子系统中唯一的一个模块，in－1 和 in－2 表示子系统的输入端口，out－1 表示子系统的输出端口(图 6－48)。

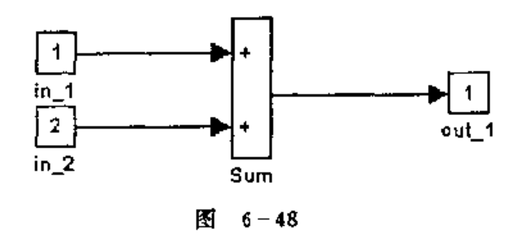


图 6－48

三、模块转变为子系统的方法

如果用户创建完了一些模块，而用户又想把这些模块变成一个子系统，那么可采用下面的方法：

(1) 用定义一个方框的办法选中用户需要变成一个子系统的模块和连线，其详细方法可参见 6.3.1 之二中的(2)"用方框选择一个以上对象"。注意不能用逐个选中它们的办法来对这些模块创建子系统。例如，图 6－49 这个模型代表一个加法器，其中在方框中的 Sum 模块和 Unit Delay 模块被选中。

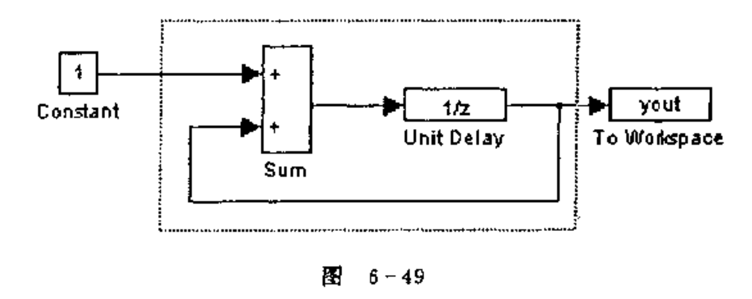


图 6－49

当用户释放鼠标按钮的时候，这两个模块和它们之间的所有连线都被选中。

(2) 在 Option 菜单中选择 Group 命令。SIMULINK 就用一个 Subsystem 模块来代替上面这两个模块和它们之间的连线。图 6－50 是在选择了 Group 命令之后所显示的模型。

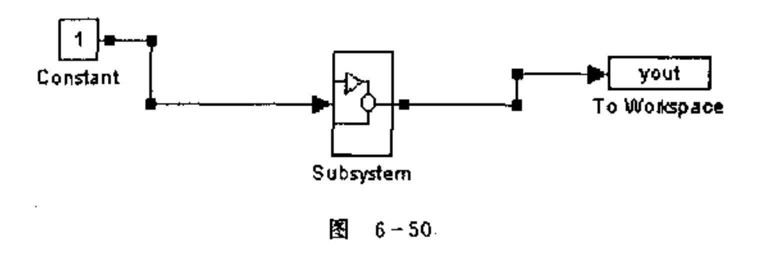


图 6－50

和所有其它的模块一样，Subsystem 模块的名字也可以改变。同样，也可以用 SIMULINK 的封装功能来定做其图标和对话框，有关其详细的内容参见第八章。

如果用鼠标双击 Subsystem 模块，SIMULINK 就显示其基本模型。图 6－51 就是在打开前面那个模型中的 Subsystem 模块后所得的结果。

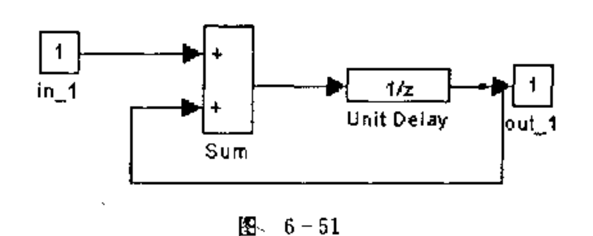


图 6-51

6.3.7 模拟方程

对于一个 SIMULINK 的新用户来说，最费解的问题之一是如何来模拟方程。下面的一些例子将帮助你理解这个问题。

一、把摄氏温度转变为华氏温度

首先来模拟一个将摄氏温度转变为华氏温度的方程：$T_F=9/5(T_c)+32$

1. 建模所需模块的确定

在进行模拟以前，首先需要确定建立上面这个模型所需要的模块：

(1) 一个 Gain 模块，用来定义常数增益 9/5，Gain 模块来源于 Linear library。

(2) 一个 Constant 模块，用来定义一个常数 32，Constant 模块来源于 Source library。

(3) 一个 Sum 模块，用来把两项相加，Sum 模块来源于 Linear library。

(4) 一个 Sine Wave 模块，用来作为输入信号，Sine Wave 模块来源于 Source library。

(5) 一个 Scope 模块，用来显示输出，Scope 模块来源于 Sinks library。

2. 模块的拷贝

把上面这些模块从各自相应的库中拷贝到用户的模形窗口中，如图 6-52 所示。

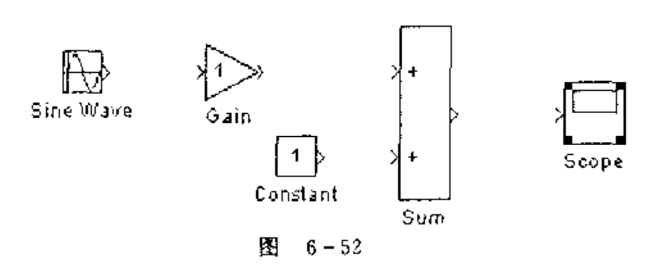


图 6-52

分别打开 Gain 模块和 Constant 模块(方法是用鼠标在它们上面双击)，分别把它们设为 9/5 和 32。然后，单击 OK 按钮。打开 Sine Wave 模块，把其幅值设为 10，以便得到较大的温度变化。

3. 模块的连接

把各个模块连接起来，得到图 6-53 所示的方块图。

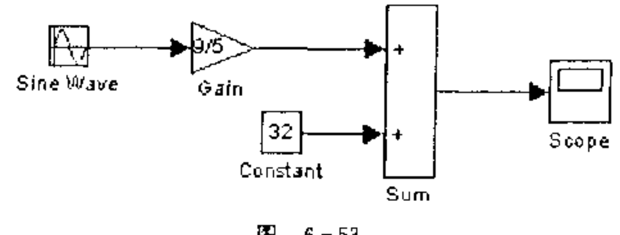


图 6-53

Sine Wave 模块代表摄氏温度，Gain 模块输出为 9/5，这个值在 Sum 模块中和 Constant 模块中的常数 32 相加后输出，这个输出就是华氏温度。打开 Scope 模块就可以观看这个输出值的变化曲线，其中，Scope 模块的 x 轴设为比较小的时间，譬如 10 s，而把 y 轴设置得比输出幅值略微大一些，以便能看到整个曲线，因此可选为 60。

4. 在用户窗口的菜单中选项

在用户窗口的 Simulation 菜单中选择 Parameters 选项，定义 Stop time 为 10 s，Maximum step size 为 0.1 s。然后在 Simulation 菜单中选择 Start 命令，仿真就开始。

二、模拟一个简单的连续系统

下面我们再来模拟一个微分方程：

$$\dot{x}=-2x+u$$

在这个模型中，需要对 $\dot{x}$ 进行积分，并产生输出，因此需要一个 Integrator 模块。另外需要 Gain 模块、Sum 模块。为了产生一个输入信号 u，可采用 Signal Generator 模块来产生方波。除此之外，还需要一个 Scope 模块来观看模型的输出。把这些模块按前面的所述的方法从各自的库中拷贝到模型窗口中，并用连线连接起来，再把 Gain 模块的增益设为 2，结果如图 6－54 所示。

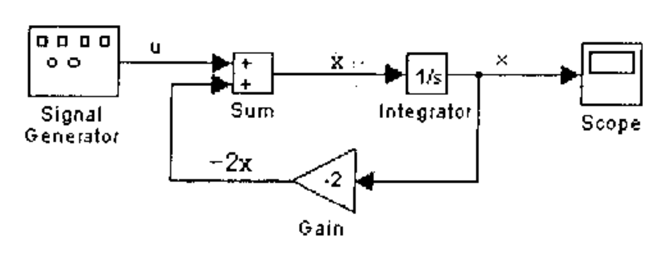


图 6－54

在这个模型中，可采用 Options 菜单中的 Flip Horizontal 选项来改变 Gain 块的方向。另外，把连线从 Integrator 块的输出端口连到 Gain 块的输入端口的时候，在画线的同时，要按下 Ctrl 键。有关其详细的内容请参见 6.3.3 之一中的(2)“从其它连线画连线”。

在这个模型中，最重要的概念是包括 Sum 块、Integrator 块和 Gain 块的那个闭环回路。在这个回路中，x 是 Integrator 块的输出，同时，$\dot{x}$ 是 integrator 块的输入且依赖于 x。因此，只有构成闭环回路，才能算出 x 的值。因此 x 和 $\dot{x}$ 之间的这种关系必须构成一个闭环才能实现。仿真时，仿真每计算一步，Scope 块就显示一步 x 的值。如果仿真时间设为 10 s，Scope 块的 y 轴范围为 1，那么其输出如图 6－55 所示。

在这个例子中，所模拟的方程也可用传递函数的形式来表示，模拟就采用 Transfer Fcn 模块，u 作为该块的输入，x 作为该块的输出。因为 Tansfer Fcn 实现的系统为 x/u，因此对该微分方程进行拉氏变换，可得

$$sx=-2x+u$$

由此可得

$$x=u/(s+2)$$

因此

$$x/u=1/(s+2)$$

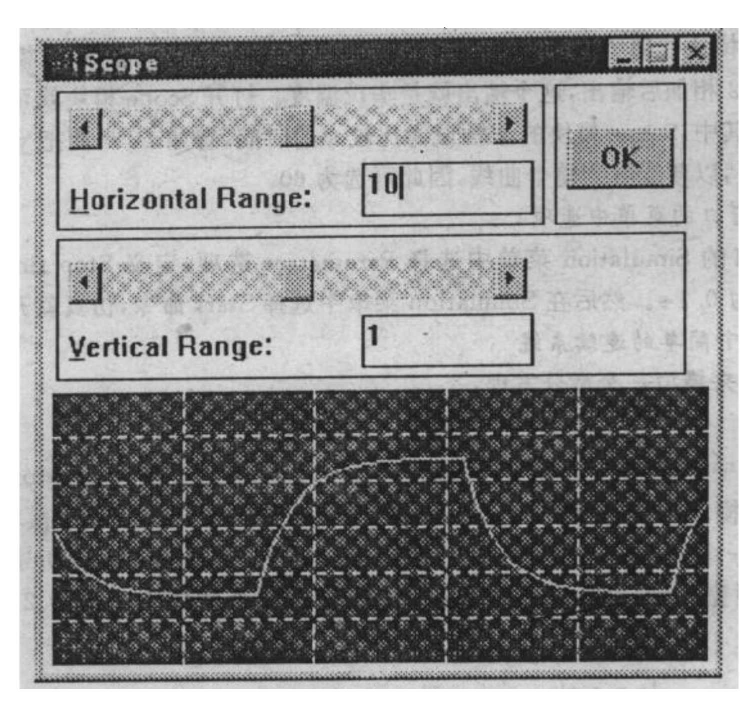


图 6-55

即传递函数为1/(s+2)，其中分子为1，分母为s+2，打开Transfer Fcn模块的对话框，定义参数numerator为[1]，denominator为[1 2]，这样模型可变为图6-56所示。

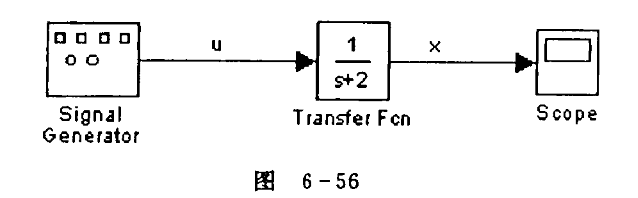


图 6-56

仿真的结果和前面的模型完全相同。

6.4 保存模型

用户可以在模型窗口中选择Save命令或Save as命令来保存模型。模型保存后，SIMULINK就生成一个M文件，这个M文件包括需要重新创建这个模型所需要的MATLAB命令。

如果用户是第一次保存模型，可用Save命令来给文件命名和指定其位置。在对话框指定位置和名字后，用鼠标单击OK按钮，就可把模型保存起来(图6-57)。

如果用户保存一个模型，而这个模型的M文件已经有了，那么使用Save命令就把原来的M文件给覆盖了，而用Save as命令可把模型保存到一个新的文件名或一个新的位置中去。Save命令是破坏性的，即选择这个命令将破坏模型前面的版本。

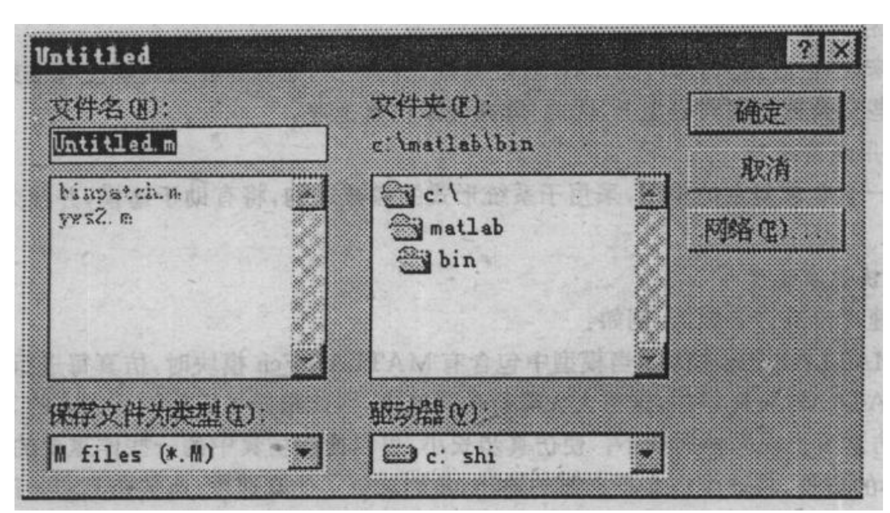


图 6-57

6.5 打印框图

SIMULINK 有两种方法可以打印框图：一种方法是在 File 菜单中选择 Print 命令；另一种方法是在 MATLAB 的命令窗口键入 Print 命令。

在 File 菜单中选择 Print 命令，打印的是当前活动窗口的框图，而不打印任何打开的 Scope 模块。

在 MATLAB 的命令窗口用 Print 命令打印指定的文件到打印机或文件中。这个文件也不打印任何打开的 Scope 模块，Print 命令的格式为：

print —smodel —ddrive filename

其中这些参数的含义为：

model：要打印的文件名字。如果省略，则打印当前系统，这时系统必须打开。

device：Windows 设备，包括：

win	当前安装的打印机，以单色打印。
winc	当前安装的打印机，以彩色打印。
meta	剪切板，以 M 文件形式。
bitmap	剪切板，以位图形式。
setup	打开 Print Setup 对话框，但不打印。

filename：保存输出的文件名。如果指定文件名，则把输出写到指定的文件中。如果文件已经存在了，则改写这个文件。如果指定的文件不包括扩展名，则添加一个适当的扩展名。如果直接把文件打印输出到打印机上，那么就不能控制其大小。如果要控制打印输出的大小，最好把打印输出先送到一个位图中，然后用 Word 程序来改变其大小。

有关 Print 命令的详细情况，请参见 MATLAB 中的 Print 命令。

6.6 建模时的一些注意事项

为了更好地使用 SIMULINK，这里提出几点注意事项：

1. 内存问题

一般来说，内存越大，SIMULINK 性能更好。内存少，意味着在仿真还没有结束的时候就需要把一些交换数据写到磁盘上，因此，就减慢了仿真速度。

2. 采用递阶结构

对于一个非常复杂的模型，采用子系统形式的递减结构，将有助于建模，并使人更容易阅读和理解。

3. 加快仿真速度

仿真速度慢有许多原因，例如：

(1) MATLAB Fcn 模块。当模型中包含有 MATLAB Fcn 模块时，仿真每进行一步，就需要调用 MATLAB 解释程序，这将大大降低仿真速度，因此最好尽可能使用固有的 Fcn 模块。

(2) 仿真步长或采样周期小。使仿真步长小，可以捕捉仿真中的一些作重要的一些细节。然而，太小的步长，将产生许多不必要的输出，从而降低了仿真速度。有关仿真步长问题将在第七章中详细讨论。

(3) 代数环。代数环的解是每步迭代进行的，因此，这将严重地降低了仿真速度。有关代数环更详细的情况，请参见第七章的有关内容。

(4) 不要把 Random Number 模块的输出送给 Integrator 模块。对于连续系统可用 Source library 中的 Band - Limited White Noise 模块来代替 Random Number 模块。

4. 模型的整理

清晰整齐的模型有助于阅读和理解。除此之外，文本注释有助于解释模型中发生了什么。有关添加文本注释的方法请参见前面有关的章节。

5. 建模方法

一般来说，建模时，首先在纸上进行，然后在计算机上进行。只有当你用 SIMULINK 把所有的模块都拷贝到模型窗口以后，才可把这些模块连起来。采用这种方法，将有助于减少打开库所需要的时间。

第七章 仿真与分析

本章主要讨论模型的仿真与结果的分析这两个部分，主要包括以下一些内容：

(1) SIMULINK 的工作方式。

(2) 仿真。

(3) 线性化。

(4) 平衡点确定。

在"仿真"这一小节将讨论如何从 Simulation 菜单和 MATLAB 命令窗口定义仿真参数和启动一个仿真的问题，同时还讨论几种不同的积分方法。

在"分析"这一小节将讨论 SIMULINK 提供的用于线性化的工具和平衡点的确定问题。

7.1 SIMULINK 的工作方式

在 SIMULINK 的模型中，每一个模块都有一些公共的特性，如图 7－1 所示：一组输入 u，一组输出 y，一组状态 x。

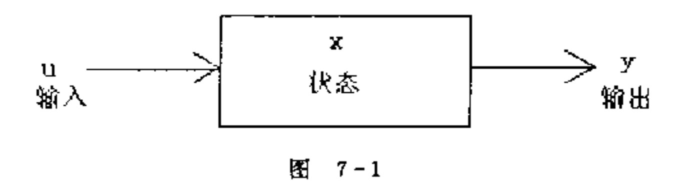


图 7－1

状态向量可能由连续状态、离散状态或两种混合状态组成，这些量之间的数学关系可以用下面的这些方程来表示：

$$y=f_0(t, x, u) \quad (输出)$$

$$x_{d_{k+1}}=f_u(t, x, u) \quad (更新)$$

$$\dot{x}_C=f_d(t, x, u) \quad (导数)$$

其中：$x=\begin{bmatrix} x_C \\ x_{d_k} \end{bmatrix}$。

仿真包括两个阶段：初始化阶段和仿真阶段。在仿真初始化阶段，一般有下面几个步骤：

(1) 所有模块的参数被传送给 MATLAB，以便 MATLAB 进行计算，并把这个最终的计算值作为各个模块真正的参数。

(2) 模型的递阶结构被展开，各个子系统模块被各子系统所包含的对象所替代。

(3) 所有的模块在仿真时，按照它们每步更新的先后顺序进行排序。排序算法保证直接向前传递的模块，只有它们的所有输入全部更新后，它们自己才能进行更新。只有在这一步，才进行代数环的检测。有关代数环更详细的情况，请参见 7.1.1 的内容。

(4) 对模块与模块之间的连接进行检查，以便确保每个模块输出的向量维数和输入的维数相同。

只有在上面的步骤完毕以后，才能进行仿真，仿真时采用数学迭代的方法进行。每种积分方法的好坏，取决于模型提供连续状态导数的能力。计算这些导数分两步：第一步，每个模块按照排序时的顺序计算各自的输出；第二步，每个模块根据当前的时间、输入和状态计算各自的导数，最终的导数向量被返回给积分器，积分器用它来计算一个新的状态向量。一旦计算出了一个新的状态向量，那么采样数据模块和图形显示模块就进行更新。

7.1.1 代数环

当由两个或两个以上对它们各自的输入直接进行前馈的模块组成反馈环时，就出现代数环。当出现代数环时，SIMULINK 就必须每步完成迭代，以便确定这个问题是否有解，这样就严重地降低了仿真的速度，也可能使问题无解。因此，应尽可能地避免出现代数环。

对输入直接进行前馈的模块有：

(1) Gain 模块。

(2) 大多数非线性模块(如：Look - Up Table、Rate Limiter)。

(3) Transfer Fcn 模块，分子分母同阶时出现。

(4) Zero - Pole 模块，当极点数和零点数相等时出现。

(5) State - Space 模块，当矩阵 D 不为零时出现。

下面一个简单的例子，用来演示一个具有代数环的系统，如图 7 - 2 所示。这个环由 Sum、Transfer Fcn 和 Gain 模块组成。

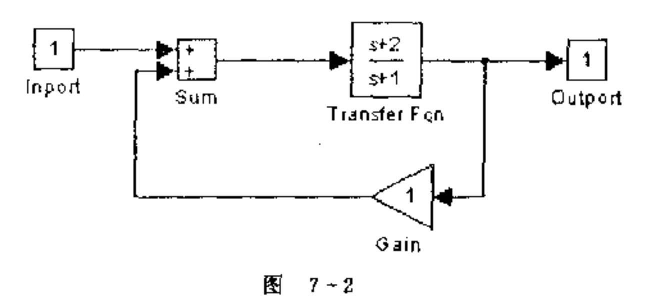


图 7 - 2

SIMULINK 在用牛顿—拉夫逊算法进行 200 迭代而这个代数环无解后，就给出一个出错信息。要切断代数环，而不是让 SIMULINK 进行迭代求解，可在代数环的任何两个模块之间插入一个 Memory 模块即可。Memory 模块具有一步积分延迟的功能，当被用来切断代数环时，总是只对环路里的第一个模块进行计算。同样，也可以用 Fransfer Fcn 模块或 Zero - Pole 模块来切断代数环。如果 Transfer Fcn 模块属于环路的一部分，用户可以在环路中加入一个小幅度高频率的极点来切断代数环。

7.2 仿 真

要启动一个仿真，有两种选择方法：一种是在 Simulation 菜单中选择命令；一种是在 MATLAB 命令窗口键入命令。

(1) 在菜单下选择命令，学起来容易，用起来简单，而且可以很快得到结果，用户也可以用屏幕观察仿真的结果。当用户创建或调试一个系统时，用这种方法比较方便。

(2) 在 MATLAB 命令窗口键入仿真和分析命令，可以使用户便于观看改变模块或者积分参数后变化的结果。

上面这两种方法在模型的不同发展阶段具有各自不同的用途。

7.2.1 仿真参数

在进行仿真以前,用户可以定义仿真参数和选择积分方法。仿真参数包括:

(1) 开始时间和结束时间。

(2) 最小和最大积分步长。

(3) 容许误差。

(4) 返回变量(如果需要的话)。

当用户用菜单命令方式进行仿真时,可以打开 Simulation 菜单中的 Parameters 子菜单,通过 Control Panel(控制面板)来定义仿真参数,然后选择一种积分方法。

当用户用命令窗口来进行仿真时,用户可以采用调用积分方法的命令来定义参数,然后进行仿真。

一、开始时间和结束时间

参数 Start Time 和 Stop Time 定义了仿真开始的时间和仿真结束的时间。仿真时间和仿真所用的时间是两个不同的概念。要进行一个 10 s 的仿真,通常计算机运行的时间不为 10 s。实际上,运行一个仿真所需的时间取决于很多因素,包括模型的复杂性、最大和最小步长、计算机时钟的频率,等等。

二、最小步长

参数 Minimum Step Size 是在仿真开始时所用的积分步长。当要产生一个输出的时候,一般来说积分器所采用的步长都不会小于这个参数所规定的值。除非是模型中包含有离散模块,而且该模块的采样周期小于最小步长。一个输出点是由 Sink 子模型库中的 Scope 模块、To Workspace 模块产生的点,或者是在一个状态或输出轨迹中返回的值。只有当积分方法进行完最后的迭代后才生成输出点。

一般来说,参数 Minimum Step Size 应该设为一个比较小的值(例如 1e - 6)。不过,当系统不连续的时候,把这个参数设得过小,就有可能在不连续处产生许多点,从而可能超过系统可用的内存和可用资源的要求。另一方面,把这个值设得过大,可能导致仿真结果不精确,而且可能错过一些重要的事件。

对于 adams 和 gear 这两种积分方法,参数 Minimum Step Size 不影响结果的精度,但是它影响所产生的输出点的个数。对于这些积分方法,最好把最小积分步长和最大积分步长设为系统便于打印或分析所需要输出点的个数所对应的那个值。

三、最大积分步长

如果把最大积分步长设置得足够小,则不易错过一些重要的细节。而一个相对比较大的积分步长,则有可能使某些模型变得不稳定。

有时,一个仿真所产生的结果非常精确,但是产生的点不平滑。在这种情况下,就有必要限制最大积分步长的大小,以便得到平滑的结果。例如:当系统是线性,而输入是分段线性的时候,对 linsim 就有必要限制其积分步长。因为,这种方法能够取任意大的步长,但精度随步长的增大而降低。rk45 方法也可以取较大的积分步长,这样就可以减少打印点的个数。

四、容许误差

参数 Tolerance 是在每步积分时用来控制积分的误差。一般来说,这个参数应该在 0.1 到 1e - 6 范围内。这个值越小,就需要更多的积分步数,而产生的结果也就越精确。然而,当容许

误差设得非常小的时候(例如 1e-10),有可能使积分步长非常小,而舍入误差却大大增加。

五、返回变量

返回变量是 SIMULINK 用来把时间、状态和输出轨迹写到工作空间里而定义的变量。在这个域中可以定义变量名。第一个变量保存的是时间,第二个变量保存的是状态,第三个变量保存的是输出。

7.2.2 用菜单进行仿真

用菜单进行仿真可允许用户在仿真期间完成一些交互的操作:

(1) 改变一个模块的参数。假定这种改变不影响该模块的状态、输入和输出的个数。

(2) 可以改变除了返回变量和开始时间以外的任何仿真参数。

(3) 改变仿真方法。

(4) 同时仿真另一个系统。

(5) 在某个连线上用鼠标单击一下,就可以把该连线上的信号送到悬浮的 Scope 模块中进行输出。

在仿真期间改变模型的结构,例如增加或删除某个连线或模块,将终止仿真的继续进行。用户也可以通过重新选择 Start 命令来观看改变后的结果。

仿真参数可以通过 Simulation 菜单中的 Parameters 来设置,SIMULINK 就显示 Control Panel 对话框,如图 7-3 所示。

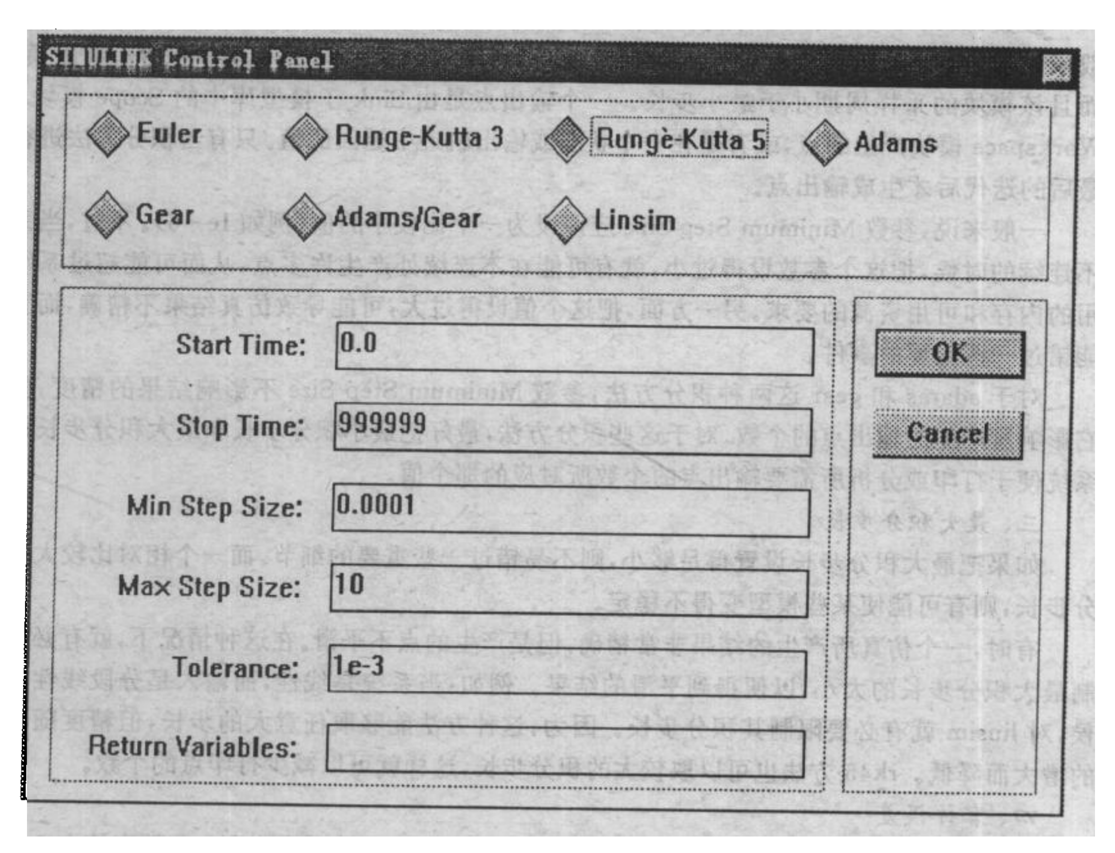


图 7-3

这个对话框可允许用户选择积分方法和定义仿真参数。有些参数可以用数值、变量名或者是有效的 MATLAB 表达式来定义。如果使用变量来定义，SIMULINK 可从工作空间上获得其值。积分方法将在下面叙述，默认的积分方法是五阶的龙格库塔法。

当定义完了所有的仿真参数，就可以进行仿真，这可以从 Simulation 菜单中选择 Start 命令。要启动一个仿真，也可通过键盘上的热键 Ctrl－T 来实现。当仿真开始以后，Start 菜单就变成了 Stop 菜单。

如果模型中有 Scope 模块，要显示其输出的结果，首先必须把它们打开。而对于 Auto－Scale Graph Scope、Graph Scope、X－Y Graph Scope 模块，SIMULINK 却自动把它们打开。

如果要终止一个仿真，可在 Simulation 菜单中选择 Stop 命令即可，也可采用键盘上的热键 Ctrl－T 来实现。

如果需要把一个仿真悬挂起来，可在 Simulation 菜单中选择 Pause 命令。如果需要把一个悬挂起来的仿真继续进行下去，可在 Simulation 菜单中选择 Continue 命令。

7.2.3 用命令进行仿真

用命令进行仿真比用菜单进行仿真具有以下一些优点：

(1) 可以不理睬模块中的初始条件(参数 x0)。

(2) 如果对用来启动仿真命令，其左边的任何选项不作定义的话，MATLAB 自动打印其输出；如果系统没有输出的话，则打印其状态。

(3) 可以定义任何外部输入(用参数 ut)。

(4) 可以由一个 M 文件来启动一个仿真，并且允许模块中的参数发生改变。

(5) 仿真可以稍微进行得快一些。

用来启动仿真的命令，其一般格式为：

```
[t, x, y]=method('model', tfinal, x0, options, ut);
```

其中只有 method、model、tfinal 这三个选项是必须的。有关这条命令的详细描述请参见第十章的“积分方法”这一节。

一、定义模块的初始条件

一开始被应用到系统的初始条件是在各个模块中定义的参数，也可以通过一个另外的向量 x0 来忽视这些初始条件的设置。但是，如果用菜单方式进行仿真，那么就不能不理睬这些初始条件。

```
[t, x, y]=method('model', tfinal, x0);
```

当向量 x0 是空([ ])或者没有定义的时侯，那么就用模块中的参数作为初始条件。

下面这条命令也可用来确定模型的初始条件。

```
[sizes, x0]=model([ ], [ ], [ ], 0);
```

下面这条命令，在其左边选项中加入第三项后，就可以获得状态的阶数。

```
[sizes, x0, xstr]=model([ ], [ ], [ ], 0);
```

其中 xstr 是一个字符串矩阵，它的第 i 行包含有与第 i 个状态(和第 i 个初始条件)有关的模块其路径全名。

例如，下面这条命令可以获得 vdp 模型初始条件的值和状态及初始条件的阶数。

```
[sizes, x0, xstr]=vdp([ ], [ ], [ ], 0);
x0 =0.250 0
```

```
        0.250 0
    xstr =/vdp/int x2
          /vdp/int x1
```

要显示 vdp 模型中的 Integrator 模块的模块名，首先必须选中 Integrator 模块，然后在 Style 菜单中选择 Title 子菜单，并选择 Displayed 选项即可。要改变模块名为 int x1 的 Integrator 模块的初始条件，只需键入下面这条命令：

```
x0(2) = 0.50;
[t, x, y] = rk45('vdp', tfinal, x0);
```

这条命令采用了 rk45 积分方法。

二、传递函数的初始条件

传递函数的模块对话框并没有提供相应的初始条件参数来供用户进行设置，原因是因为按照状态变量的形式，传递函数的表达不唯一。要设置这些模块的初始条件，可以在调用积分函数的时侯，用变量 x0。在这种情况下，代表传递函数的状态被变成了状态空间伴随形式，就像用 tf2ss.m 函数所返回的那样。

7.2.4 观看输出轨迹

SIMULINK 可以有三种方法来显示输出轨迹，分别是：

(1) Scope 模块。

(2) 返回变量和 MATLAB 的画图命令。

(3) To Workspace 模块和 MATLAB 的画图命令。

一、采用 Scope 模块

Scope 模块可在仿真进行的同时用来显示输出轨迹。下面这个简单的模型就用来演示 Scope 模块的使用方法，如图 7-4 所示。

在 Scope 模块中进行输出显示是非常原始的，它只显示输出轨迹，而没有任何标记。而其它一些 Graph Scope 模块，像 Auto-Scale Graph Scope、Graph Scope、XY Graph Scope 模块，除了显然输出轨迹外，还显示两个轴和彩色的线型，但是执行的速度要比 Scope 模块慢得多。

二、采用返回变量

根据返回的时间值和输出值，就可以用 MATLAB 的画图命令来显示和标记输出轨迹，如图 7-5 所示的模型。

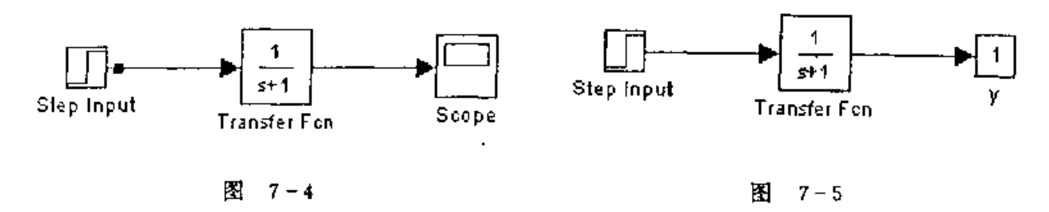


图 7-4　　　　图 7-5

名字为 y 的模块是一个在 Connections library 中的 Output 模块。输出轨迹 y 由积分函数返回。假如把系统命名为 tfout，这样可从命令行中调用仿真：

```
[t, x, y] = linsim('tfout', 2);
```

这样就可生成时间 t 和输出 y。也可在 Simulation 菜单中进行仿真，只要把参数 Return Variable 定义为[t, x, y]，然后就可以用下面的命令来画图：

plot(t, y);

三、采用 To Workspace 模块

To Workspace 模块的功能是把输出轨迹放到 MATLAB 的工作空间里的变量中。下图 7－6所示的模型演示的就是 To Workspace 模块的这种功能。

当仿真结束以后，变量 y 和 t 就存放在工作空间里。时间向量是通过把 Clock 模块的输出送给 To Workspace 模块来保存的。对于用菜单进行驱动的仿真，时间向量也可通过参数 Return Variable 域，在域中键入 t 来获得，也可由下面的积分函数的返回值来得到：

t = linsim('tfout', 2);

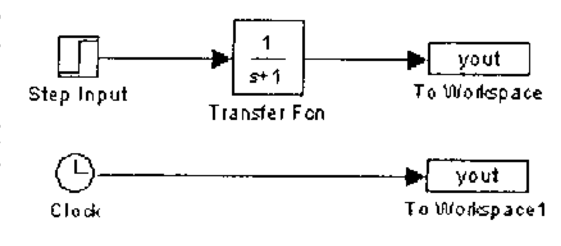


图 7－6

To Workspace 模块可接受一个向量输入，每个输入元素的轨迹以一个列向量的形式存放在工作空间的变量中。

7.2.5 积分方法

SIMULINK 模型的仿真要涉及到一组常微分方程的数值积分。因此，SIMULINK 就为这些方程的仿真提供了许多积分方法，如表 7－1 所示。

表 7－1

方 法	说 明
linsim	提取线性动态特性的方法
rk23	三阶龙格库塔方法
rk45	五阶龙格库塔方法
gear	用于刚性系统的预测—校正法
adams	亚当斯预测—校正法
euler	欧拉法

一、积分方法的选择

由于系统动态特性的多样性，因此，没有一种方法能够精确而且有效地适用于各种模型的仿真。要获得快速、准确的仿真结果，就必须精心选择适当的方法和适当的仿真参数。

对于不同的模型和不同的条件，仿真的性能(如仿真的速度、精度等)是千变万化的。因此，在选择仿真方法时，必须时刻牢记这些差异。

1. linsim

linsim 是一种用来提取系统线性动态特性的方法。被仿真的系统越接近于线性时，这种方法的性能就越好。对于由 Tansfer Fcn、State－Space、Zero－Pole、Sum 和 Gain 这些模块组成的线性模型，linsim 可采用比较大的积分步长。因此要获得合适的输出个数，就必须对最大步长进行限制。

当系统中的线性模块既有快时变，又有慢时变动态特性(即刚性系统)时，linsim 比其它积

分方法要显得优越得多。

2. rk23 与 rk45

龙格库塔方法 rk23 和 rk45 是一种对于大多数问题都比较适用的方法。当系统有高度非线性或者不连续性时，这些方法比其它方法通常要显得优越。但是这种方法不适于既具有快时变，又具有慢时变动态特性的系统，即刚性系统。在这种情况下，可以用 linsim 或者 gear 这两种方法来解决这个问题。对于连续和离散这类混合系统，rk23 和 rk45 这两种方法的性能也比较好。

尽管 rk45 与 rk23 相比，仿真速度相对更快、仿真精度相对更高。但是 rk45 输出的个数要比 rk23 少。因此要进行平滑画图时，最好采用 rk23。

3. adams 和 gear

对于状态轨迹平滑的系统，admas 和 gear 这两种预测-校正法工作性能比较好。其中，gear 这种方法是专为刚性系统设计的，而对于非刚性系统，这种方法的性能就不如其它几种方法。而对不连续系统，这种方法就不能很好地工作。

gear 这种方法可用于平滑的、非线性的系统，但是当系统中有奇异值或者系统受到快速时变信号干扰时，这种方法就不能很好地工作。

admas 这种方法可用于平滑的、非线性的、时间常数变化不大的系统。

4. euler

euler 是一种前向显式的欧拉方法。一般来说，一阶的方法要比高阶的方法需要的时间点数多。和其它方法相比，这种方法就没有其它几种方法那么高的精度。因此应该尽量避免使用这种方法，除非你的模型需要进行实时仿真。

二、步长的控制

下面这条命令可以用来定义仿真参数：

[t, x, y]= linsim ('model', tfinal, x0, [tol, minstep, maxstep]);

在这条命令中，积分参数向量[tol, minstep, maxstep]分别定义了容许误差、最小积分步长、最大积分步长。其中：标量 minstep 和 maxstep 可用来限制积分步长的大小，使之满足 minstep ≤积分步长≤maxstep。

1. 步长固定的方法

尽管 euler、linsim、rk23 和 rk45 这几种方法，它们的步长是可变的，但也可通过设置最小步长等于其最大步长的方法，来使这些方法具有固定的步长，如：

[t, x, y] = linsim (' model ', tfinal, x0, [tol, fixstep, fixstep]);

当上面这些方法识别出最小积分步长等于最大积分步长时，就不再进行误差检查。因此，仿真速度加快了，但是精度也降低了。

由于 admas 和 gear 这两种方法在两个输出之间的点数是不定的，因此就不能对它们的步长进行固定。对这两种方法定义最大积分步长和最小积分步长，是为了确保 SIMULINK 能够捕捉到能够被步长除尽的那些点上的数据。

2. 实际步长

在每步积分时，积分函数就调用模型，来求取其状态的导数和计算其输出。对于每个输出点来说，它是由许多求取模型导数的调用来产生的，只有 linsim 和 euler 这两种方法每产生一个输出只需调用一次。而对于其它的积分方法，就有可能采用比最小积分步长更小的步长，原

因是积分步长被分段了。例如，rk45 这种方法，对于每一个输出点来说需要 6 次调用求取模型的导数，也就是说需要 6 个步长，依次是[0, 1/4, 3/8, 12/13, 1, 1/2]。而对于 rk23 这种方法，需要三个步长，分别是[0, 1/2, 1]。而 admas 和 gear 这两种预测一校正法，在每个输出点之间，其步数是可变的。

当误差超过参数 Tolerance 所规定的值时，积分器就有可能回过头来取当前以前的积分步长，特别是当系统中有非线性时，就很有可能出现这种情况。

3. 数据的插值

积分函数通过自动改变积分步长的大小来使输出点的误差在局部误差所估计的范围之内，这意味着在模型积分以后将产生间隔不等的数据点，这就给比较由不同方法进行仿真所产生的结果带来了困难，这时可采用 MATLAB 的插值函数，在你需要的时间点上进行插值，获得其估计值。

下面这个例子中所采用一个系统，用来为一个容许误差小的仿真产生其平均误差的近似值，用户可以通过 MATLAB 的插值函数 interp1 对容许误差大的仿真进行插值，然后就可以比较这两者的结果：

```
% High tolerance simulation
  [t, x, y ] = linsim ('system', 10, [ ], [1e-8, 1e-6, 1 ] );
% Low tolerance simulation
  [t1, x1, y1 ] = linsim ('system', 10, [ ], [ 0.1, 1e-5, 1 ] );
% Linearly interpolate
  [ yint ] = interp1 ( t, y, t1 );
  err_per_point = sum ( abs ( yint - y1 ) )/ length ( t1 );
```

如果需要的数据比仿真的数据多得多，这时最好采用多项式插值。例如，要进行二次型插值来添加数据点，以便进行平滑画图，可采用 MATLAB 的 interp2 函数

```
% Generate 200 points for plotting
  tfine = 0: 0.05: 10;
  [ t, x, y ] = rk45 ('system', 10, [ ], [ 1e-2, 1e-4, 1 ]);
% Interpolate over time scale
  yint = interp2 ( t, y, tfine );
  plot ( tfine, yint )
```

同样，可采用 MATLAB 的 spline 命令进行三次样条插值。

```
  yint = spline ( t, y, tfine );
```

不过，当数据集比较大时，最好不要采用这种方法，因为这种方法对内存的要求很高。

三、积分方法的比较：示例

仿真的性能取决于所选的方法和设置的参数。下面用 Van-der-Pol 方程、一个二阶的非线性模型来比较几种不同的积分方法：

$$\ddot{x}+(x^2-1)\dot{x}+x=0$$

这个方程可以用下面的一阶微分方程组来描述：

$$\dot{x}_1=x_1(1-x_2^2)-x_2$$
$$\dot{x}_2=x_1$$

由这个方程组可得到图 7－7 所示的结构框图，并把它命名为 vdp。

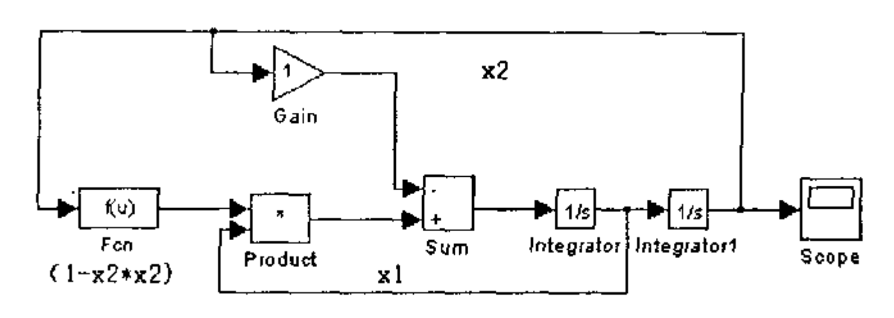


图 7－7

在这个例子中，通过定义容许误差为 1e－3 来比较几种不同的积分方法，其中的参数是这样的：

```
tol = 1e-3;
minstep = 1e-5;
maxstep = 10;
options = [ tol, minstep, maxstep ];
x0 = [ 1; 1 ];
```

进行比较的积分方法有 linsim、rk23 和 rk45：

```
[ tls, xls ] = linsim ( 'vdp ', 10, x0, options );
[ tr23, xr23 ]= rk23 (' vdp ', 10, x0, options );
[ tr45, xr45]= rk45 (' vdp ', 10, x0, options );
plot ( tls, xls, tr23, xr23, ' o ', tr45, xr45, ' + ');
```

三个状态的轨迹曲线如图 7－8 所示。

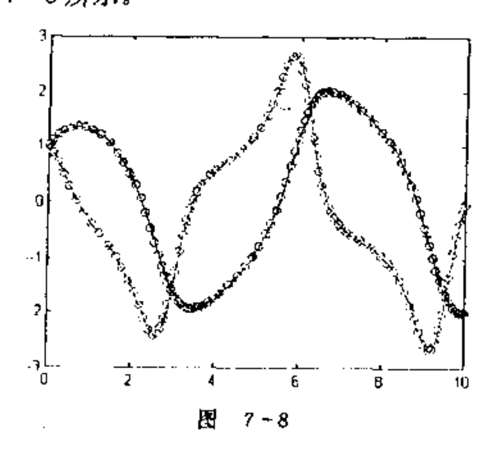


图 7－8

从图 7－8 中可以看出：三个结果是非常吻合的。如果用 admas 和 gear 方法与其它的方法进行比较，也会发现它们的结果与其它的方法的结果也是一致的。表 7－2 是在容许误差为 1e－3 时的对比结果。

表 7-2

方 法	求导次数	输出点数	执行时间(比率)	每点误差
linsim	144	136	1.37	0.006 1
rk23	316	89	1.60	0.002 3
rk45	204	35	1	0.004 1
adams	270	55	1.51	0.001 9
gear	336	65	1.91	0.004 9
euler	365	366	3.03	0.288 1

下面再来比较一下容许误差为 0.1 时的各种不同的积分方法：

```
options(1) = 0.1;
```

同样，被比较的方法有 linsim、rk23 和 rk45：

```
[ tls, xls ] = linsim ('vdp', 10, x0, options );
[ tr23, xr23 ] = rk23('vdp', 10, x0, options );
[ tr45, xr45] = rk45('vdp', 10, x0, options);
```

这时，可用下面的命令来把这三个状态轨迹曲线和容许误差为 1e-6 的 rk45 方法画在同一张图 7-9 上进行比较：

```
[ t, x ] = rk45 ('vdp', 10, x0, [ 1e-6, 1e-5, 10 ]);
plot ( t, x, tls, xls, 'x', tr23, xr23, '-', tr45, xr45, ':')
```

从图 7-9 中可以看出：响应的性能明显降低，其中 linsim 的响应性能最好，而用 rk45 产生的响应性能损失最大。

下面再来比较一下容许误差为 0.1 时，adams、gear 和 euler 方法：

```
[ ta, xa ] = adams ('vdp', 10, x0, options );
[ tg, xg ]= gear ('vdp', 10, x0, options );
[ te, xe ]= euler ('vdp', 10, x0, options );
plot ( t, x, ta, xa, 'x', ta, xg, '-', te, xe, ':')
```

如图 7-10 所示。

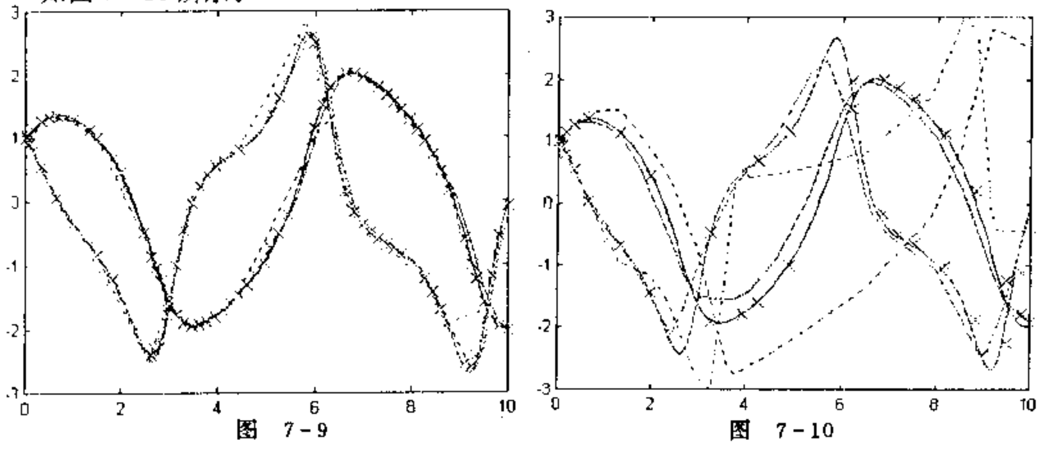


图 7-9　　图 7-10

从图 7－10 中可以看出，euler 方法不稳定，而 admas 和 gear 方法偏离希望的响应曲线。在容许误差为 0.1 时，各种方法求导的次数如表 7－3 所示。

表 7－3

方　法	求导次数	输出点数	执行时间(比率)	每点误差
linsim	77	74	2.38	0.032 6
rk23	86	23	1.29	0.373 0
rk45	140	21	1.46	0.881 1
adams	116	34	2.69	0.100 5
gear	181	48	4.17	0.231 9
euler	33	34	1	1.583 5

7.2.6　离散时间系统

SIMULINK 具有对离散时间系统进行仿真的能力。模型可以是多采样速率的，即模型中包含有不同采样速率的模块。模型既可以由纯粹的离散模块来组成，也可以由混合模块来组成，即模型中既含有离散模块，也可以含有连续模块。

一、离散模块

每一个离散模块在输入端口都有一个采样器，而在输出端口都有一个零阶保持器。当离散模块和连续模块混合在一起的时候，离散模块在两个采样点之间的输出保持常值。即离散模块的输入只有当采样时间到的时候才进行更新。

二、采样时间

参数 Sample time 用来设置离散模块的采样时间，这个采样时间也是离散模块的状态进行更新的时间。一般来说，采样时间被定义为标量，也可在这个域中定义一个具有两个元素的向量，其中一个元素表示偏移时间。

例如，定义参数 Sample time 为[Ts，offset]，则采样时间为 Ts，偏移时间为 offset，那么离散模块只在采样时间的整数倍加上 offset 这个时间才进行更新，即

$$t=n*Ts+offset$$

其中 n 是整数，offset 既可以是正数，也可以是负数，但是不能大于等于采样时间。如果某些模块更新得比其它模块早一些或晚一些的时候，这时 offset 就非常有用。当仿真正在进行的时候，每个模块的采样时间不能进行更改。如果确实需要改变其当前值，只有暂时终止仿真，修改完后，再重新进行仿真即可。

三、纯粹的离散系统

对于纯粹的离散系统可以用任何积分方法进行仿真，而且仿真的结果没有任何区别。如果要想得到采样时间时刻的输出，只需把最小积分步长设置得比最大采样时间大就可以了。

四、多采样率系统

多采样速率系统包含有以不同采样速率进行采样的模块，这些系统既可以用离散模块来模拟，也可用离散模块和连续模块混合形式来模拟。例如，一个多采样率离散模型如图 7－11 所示。

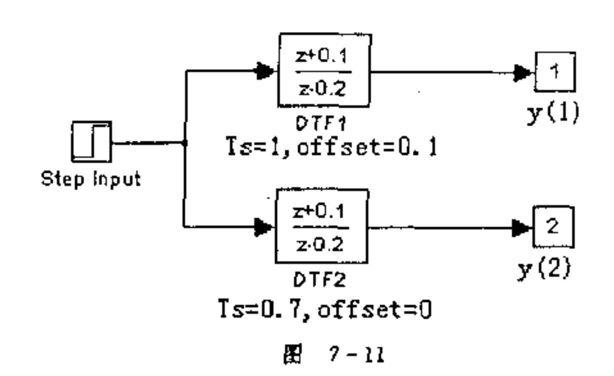


图 7-11

Step Inport 模块中的参数 Step time 可以从 1 变到 0。名字为 DTF1 的 Discrete Transfer Fcn 模块中的参数 Sample time 和 offset 分别设为 1 和 0.1，而名字为 DTf2 的 Transfer Fcn 模块中的参数 Sample time 和 offset 分别设为 0.1 和 0。用下面的命令调用仿真和进行画图：

```
[ t, x, y] = euler ( ' multirate ', 3);
stairs ( t, y )
```

所得的结果如图 7-12 所示。

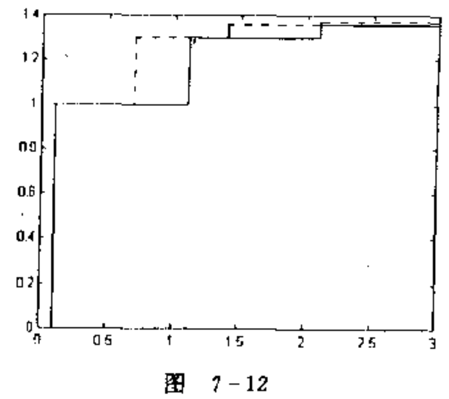


图 7-12

在这个例子中，y(1)的输出为实线，y(2)的输出为虚线。对于 DTF1 模块，由于其 offset 为 0.1，因此当 t=0.1时，该模块才能对输入进行前馈。既然传递函数的初始条件为 0，因此在 t=0.1 之前，DTF1 的输出为 0。如果要使 DTF1 在 t=0 时刻输出为 1，可把 offset 设为 -0.9 即可。

五、采样时间的颜色

采样时间的不同颜色特征可以帮助用户在模型中区分不同的采样速率，因为不同的颜色，就表示不同的采样速率。有关各种颜色的含义，将在下面的小节中叙述。

要理解这些功能是如何工作的，就有必要熟悉 SIMULINK 的采样时间传递机制。图 7-13 表示一个由采样时间为 Ts 的离散 Filter 模块驱动一个 Gain 模块的系统。由于 Gain 模块只对其输入乘上一个常数后就直接输出，因此它的采样速率和 Filter 的采样速率是一样的。换句话说，Gain 模块的有效的采样速率就等于滤波器的采样速率，这就是所谓的采样时间传递机制的基本机理。

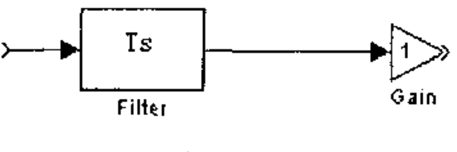


图 7-13

要使采样时间颜色特征有效，必须在 Style 菜单中选择 Sample Time Colors 命令。当用户对模型进行改变的时候，SIMULINK 不会自动对模型中的颜色进行改变。因此，要想得到改变后模型的颜色，就必须在 Style 菜单中选择 Update Diagram 命令；而要想得到原来的采样时间颜色，只需再次选择 Sample Time Colors 命令。

六、颜色的含义

当使用采样时间颜色的时候，给每一模块所指定的颜色取决于模型中各个模块的采样时

间。表 7 - 4 列出了各种颜色的含义。

表 7 - 4

颜 色	含 义
黑 色	连续模块
黄 色	混合模块
红 色	采样率最快模块
绿 色	采样率次快模块
蓝 色	采样率第三快模块
蓝绿色	采样率第四快模块

对于每一模块来说，采样时间按照下面的规则进行设置。

(1) 连续模块(如 Integrator Derivative Transfer Fcn 等)，根据定义是连续的。

(2) 离散模块(如 Zero -Order Hold Unit Delay Discrete Trancfer Fcn 等)，它们的采样时间经过各自的对话框由用户来定义。

(3) 所有其它模块的采样时间是隐含定义的，它取决于该模块输入信号的采样时间。例如，一个 Gain 模块跟在 Integrator 后面，它就是一个连续模块。反之，如果一个 Gain 模块跟在一个 Zero - Order Hold 后面，它就是一个离散模块，并且它的采样时间和 Zero - Order Hold 模块的采样时间相同。

对于具有多个输入、而各个输入的采样时间不同的模块，如果输入信号中有一个最快的采样时间，而其它信号的采样时间又是它的整数倍，那么该模块的采样时间就是那个最快输入信号的采样时间。否则的话，该模块就看成连续的模块。

如果模型中有 4 个以上的采样速率，那么采样速率最快的那 4 个模块的颜色分别为红、绿、蓝和蓝绿色，其余的都为黄色，并且 SIMULINK 会给出相应的信息。

由于 Mux 和 Demux 模块只简单地进行归并或分组运算，输入信号通过它们后仍保持它们原来的值，由于这个原因，如果它们由不同采样速率的信号驱动，它们仍然只有唯一的颜色。在这种情况下，Demux 和 Mux 模块就变成混合色(黄色)来表示它们处理的信号具有多个采样速率。同样，包含有不同采样速率的子系统模块也变成混合色。如果一个子系统模块中所有的模块具有一个相同的采样速率，那么子系统模块的颜色就根据那个采样速率来确定。

在某些不影响仿真输出的情况下，SIMULINK 也把采样时间反传给前面的驱动模块。例如，图 7 - 14 所示的模型，SIMULINK 识别出 Signal Generator 模块正在驱动一个 Discrete - Time Integrator 模块，因此 SIMULINK 就给 Signal Generator 和 Gain 模块设定和 Discrete - Time Integrator 模块相同的采样速率。

用户可以通过激活 Sample Time Colors 来证实上面的结论。实际上，当 Sample Time Colors激活后，所有的模块都变成红色，因此上面的结论成立。另外，由于 Discrete - Time Integrator 模块只有在采样时刻才查看其输入，因此上面这种改变不影响仿真的结果。如果用图 7 - 15 所示的连续 Integrator 模块代替 Discrtet - Time Integrator 模块，然后再用 Style 菜单中 Update Diagram 命令，就会发现 Signal Generator 和 Gain 模块都变成连续的模块。因此，它们

也都变成黑色。

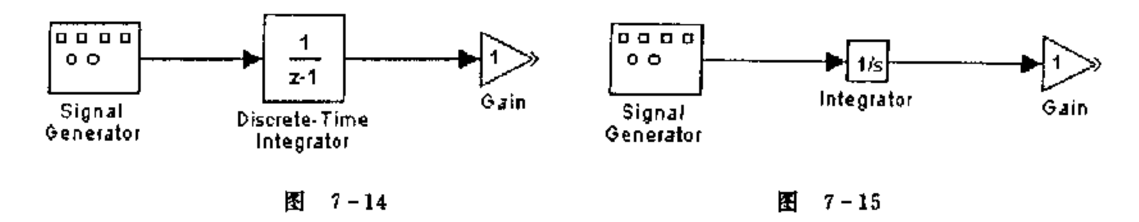


图 7-14　　　　图 7-15

七、混合系统

连续和离散这类的混合系统是由采样模块和连续模块组成的，对于这样的系统可用任何一种积分方法进行仿真，只是有的方法可能速度快些、精度高些，而有的方法可能低一些。对于连续和离散这样的混合系统，龙格库塔变步长方法 rk23 和 rk45 从速度和精度这两个方面来说，要比其它的方法要来得优越。由于离散模块采样和保持所形成的不连续性，因此 gear 和 adams 不适于这类混合系统的仿真。

7.3 线线化

SIMULINK 提供了两个基本函数 linmod 和 dlinmod 用来提取以状态空间 A、B、C 和 D 形式的线性模型。状态空间矩阵描述的是如下形式的输入和输出之间的关系：

$$\dot{x}=Ax+Bu$$

$$\dot{y}=Cx+Du$$

其中：x、u 和 y 分别为状态、输入和输出向量。例如，图 7-16 所示为一个叫作 lmod 的模型。

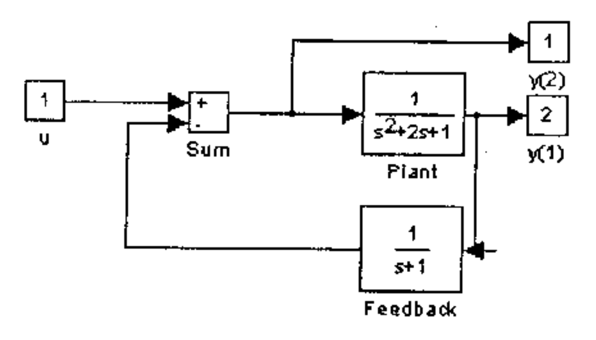


图 7-16

要提取这个 SIMULINK 模型的线性化模型，可以键入下面的命令：

[A,B,C,D] = linmod('lmod')

$$A=\begin{bmatrix}-1 & 0 & 1\\ -1 & -2 & -1\\ 0 & 1 & 0\end{bmatrix}\quad B=\begin{bmatrix}0\\ 1\\ 0\end{bmatrix}\quad C=\begin{bmatrix}0 & 0 & 1\\ -1 & 0 & 0\end{bmatrix}\quad D=\begin{bmatrix}0\\ 1\end{bmatrix}$$

系统的输入和输出必须由 Connections 库中的 Inport 和 Outpu 模块来定义，Signal Generator和 Scope 模块不能作为系统的输入和输出。一旦得到状态空间形式的数据，就可以用信号和系统工具箱中的函数对其进行进一步的分析：

(1) 转换成传递函数形式。

```
[ num, den ] = ss2tf ( A, B, C , D);
```
(2) 伯特图。
```
bode ( A, B, C, D );
```
(3) 时间响应。
```
step ( A, B, C, D );
impulse ( A, B, C, D );
lsim ( A, B, C, D, u, t );
```
其它一些函数可用来对线性系统进行控制器设计。

当模型是非线性的时候,首先必须选定一个工作点,然后在这个工作点附近提取线性模型。非线性模型对于工作点处的干扰是非常敏感的,因此必须在截断和舍入误差之间进行折衷。linmod 命令的另外一些选项分别定义了工作点和干扰点。

```
[ A, B, C, D ] = linmod ( ' model ', x, u, pert, xpert, upert )
```

对于离散系统或者离散、连续这样的混合系统,可用函数 dlinmod 进行线性化。除了右边第二个选项必须具有采样时间以外,这个命令和 linmod 一样,具有相同的调用格式。有关更详细的情况请参见第十章中对 linmod 命令的描述。

对于模型中含有 Derivative 或 Transport Delay 模块的系统,用 linmod 进行线性化是比较麻烦的。因此在线性化之前,用特殊设计的模块来替换它们,以便避免出现这种问题。这些特殊设计的模块可在 Extras library 的 Sublibrary 中找到。

7.4 平衡点的确定

SIMULINK 提供的函数 trim 可以用来确定稳态平衡点。下面以一个叫作 lmod 的模型为例,如图 7-17 所示。

在这个模型中,用户可以用 trim 函数来找到使输出为 1 时输入和状态所对应的值。首先输入 u 和状态变量 x 作为初始假设,然后设置输出 y 的希望值:

```
x = [0; 0; 0];
u = 0;
y = [1; 1];
```

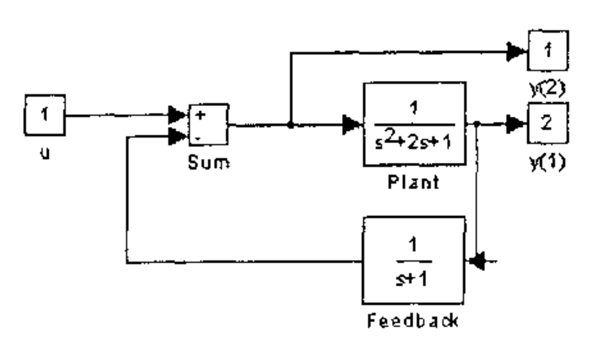


图 7-17

可用指针变量指出哪个变量是固定的,哪个变量是可以变化的,如:

```
ix = [ ];       % Don't fix any of the states
iu = [ ];       %Don't fix the input
iy = [1;2];  %Fix both output 1 and output 2
```

调用 trim 函数,返回所得的解。

```
[ x, u, y, dx] = trim ( ' lmod ', x, u, ix, iu, iy )
```

$$x = \begin{bmatrix} 1.0000 \\ 2.0000 \\ 1.0000 \end{bmatrix}$$

$$u = 2$$

$$y = \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}$$

$$dx = \begin{bmatrix} 1.0\,e{-15} \\ 0 \\ 0.4441 \\ 0.2693 \end{bmatrix}$$

由于舍入误差的影响,所得的结果可能会有所差异。

注意到有时会遇到没有平衡点问题,在这种情况下,在首先设置偏差为 0 以后,trim 函数返回的解是使最大偏差最小的那个解。trim 命令的格式将在第十章中描述。

第八章　用封装的办法来定制新模块

利用 SIMULINK 的封装功能，可以定做一个模块或一个子系统的对话框和图标。本章主要讨论以下一些问题：

(1) 封装过程的概述。

(2) 用封装的办法创建新模块。

(3) 为新模块定义图标。

8.1　封装过程概述

8.1.1　封装的作用

采用封装的办法，有下列好处：

(1) 使模型的用户与模型不必要的复杂性隔绝开来。

(2) 提供一个描述性的、有益的用户接口。

(3) 保护模块的内容免受无意干扰的影响。

对于一个具有一个模块以上的子系统来说，封装的一个重要的用途是帮助用户创建一个对话框来接受参数。因此，这样就无须打开子系统中各个模块的对话框，然后再逐个输入参数，而是把这些模块组成一个子系统，然后定义这个子系统的对话框来接受这些参数值。

8.1.2　封装的实施

一般来说，整个封装过程包括下列这些步骤：

(1) 把各个模块打开，然后给那些需要在新的对话框中进行赋值的那些参数指定一个变量名。

(2) 创建子系统。有关子系统的创建方法详见第六章。

(3) 选择子系统模块，然后在 Option 菜单中选择 Mask 选项。

(4) 填写封装对话框。在 8.2.1 节中将详细讨论这个问题。

(5) 用鼠标单击 OK 按钮来创建新模块，子系统模块所显示的图标就是封装对话框中定义的图标。如果打开了新的子系统模块，SIMULINK 就显示用户创建的新的对话框。

8.2　用封装的办法创建模块

假定我们需要创建一个二阶的子系统模块和它的图标及它的对话框，那么封装 Transfer Fcn 模块就可以让我们用变量而不是用每次仿真时定义其分子和分母的形式来定义其参数，这些参数包括自然频率(wn)和阻尼系数(zeta)。

下面用 Sine Wave 模块作为输入，并用 Transfer Fcn 模块和 Scope 模块共同来建立一个简单的模型。在 Transfer Fcn 模块的对话框中，把分子定义为[wn ^ 2]，分母定义为[1　2 * zeta * wn wn^2]。当把所有的模块连接完后，模型就如图 8-1 所示。

8.2.1　填写封装对话框

要封装 Transfer Fcn 模块，首先必须选中该模块，然后从 Options 菜单中选择 Mask 选

项，这时就显示如图 8－2 所示的封装对话框。在这个对话框中允许用户定义参数域、初始化命令、图标和帮助文本来对封装后的模块进行描述。对话框中的标题、模块名和模块的形式将从被封装的模块中继承，在这个例子中，即是 Transfer Fcn 模块。

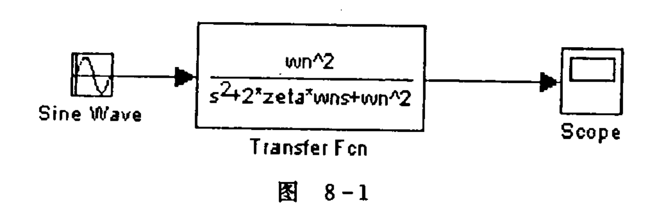


图 8－1

Transfer Fcn

Block name: Transfer Fcn
Block type: Transfer Fcn

Mask Block Definitions

OK
Cancel
Help

New block type:

Dialog strings separated by | :

Initialization commands:

Drawing commands:

Help string:

图 8－2

SIMULINK 将采用用户在这个对话框中键入的信息来创建一个新模块的对话框和图标，这些信息包括模块的形式、标题、对话框参数域的标号、模块初始化方式、模块图标的形状以及在对话框中按下 Help 按钮时所显示的文本信息。

图 8－3 演示的就是我们在这个例子中创建的对话框及其对话框的各个部分与被封装模

块对话框中各个参数域的对应关系。

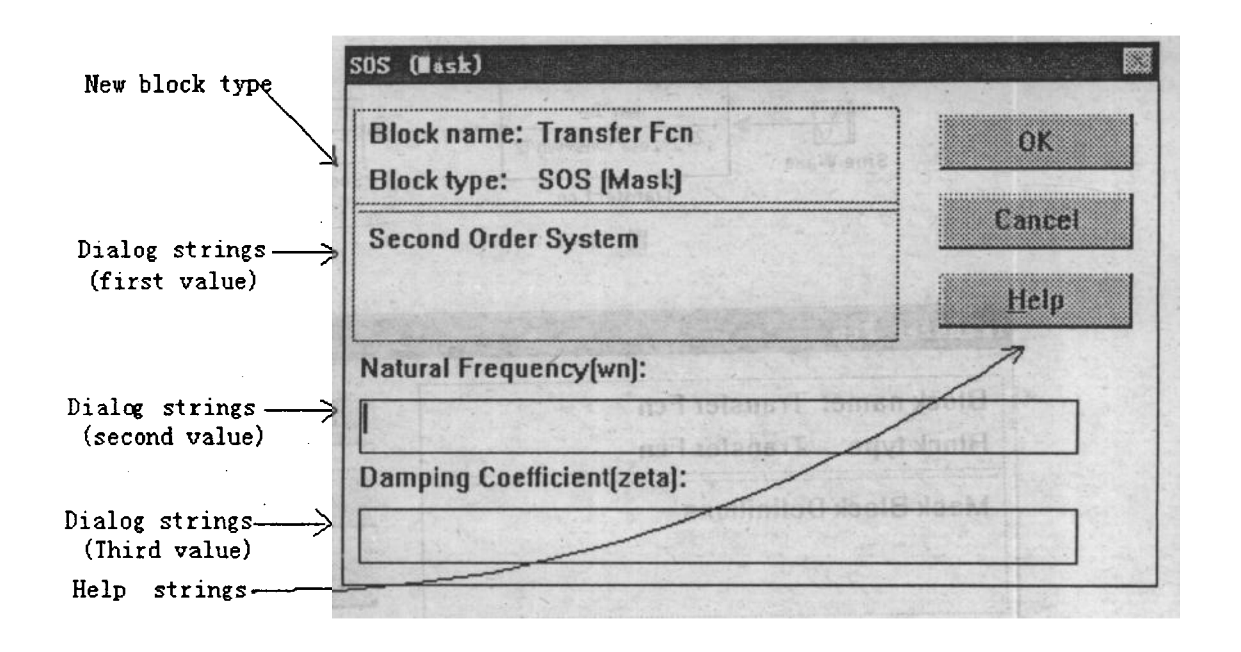


图 8-3

下面这一节将就封装对话框分别讨论其中各个参数的含义。

一、New Block Type 域

New Block Type 这个域是用来定义封装后所得模块的类型。一个模块的类型是指一个任意的对模块进行的描述，它与模块的性能没有任何关系。一个模块的类型在对话框的标题和 Block type 域中显示。

对于这个例子，键入

SOS

SIMULINK 就在 SOS 的后面跟上"(Mask)"。

二、Diaglog String 域

Diaglog String 域是用来到定义新模块对话框中的下列一些内容：

(1) 模块的描述。

(2) 参数域标号。

Diaglog String 域可以用下面的格式来定义这些项：

BlockDesc|Parameter1Label|Parameter2Label|...|ParameternLabel

中间用垂直杠"|"把各个值分开，在垂直杠的前后无需加入空格。如果在对模块的描述中强制结束一行，则添加字符\n。

用户最多可以定义 6 个参数域。如果需要定义 6 个以上参数，就必须有一个或多个参数域来接受存放有一个以上参数的向量。例如，用户可以把 Natural Frequency 和 Damp Coefficient

域合并为一个域，然后以一个向量的形式输入其值，如：

Natural Frequencey [wn] and Damping Coefficient [zeta]:

[2.0, 0.707]

如果上面定义的这个域是对话框中的第一个参数域，就可以用下面的初始化命令来对 Natural Frequency 值(2.0)进行定位：

wn = @1(1);

而对 Damping Coefficient 值(0.707)进行定位可用这样一条命令：

zeta = @1(2);

有关如何在初始化命令中对模块中参数值进行定位的详细描述将在后面叙述。

1. 定义模块的描述

模块的描述作为对话框中的一个信息显示在模块名和模块类型的下面。对于上面这个例子，键入

Second Order System

2. 定义参数域和标号

要为新对话框创建参数域，可在模块描述的后面键入它们的标号就可以了。对于上面这个例子，封装对话框为自然频率和阻尼系数提供了两个参数域，因此可在 Dialog string 域中键入：

Second Order System|Natural Frequency(wn):|Damping Coefficient(zeta):

(注意对模块的描述 Second Order System 在前面已经输入过了)

3. 定义一个 MATLAB 命令

如果用户要让 SIMULINK 调用 MATLAB 的一条命令，而不是打开模块对话框的话，就必须在域中键入那条 MATLAB 命令。

例如，在 Dialog string 域中键入下面的命令，则每当该模块被鼠标双击打开时就画出一条直线。

eval ('plot (1:10)')

三、Initialization Commands 域

1. 初始化方式

在 Initialization Commands 域中定义的是模块的初始化方式。初始化命令可以使用在新模块对话框参数域中键入的参数值。当定义初始化命令的时候，用户可以由新模块对话框中键入的值，根据其相对位置，用下面的命令来对其进行定位：

variable = @field

其中 variable 是封装后所得模块、在其对话框中定义的变量名，field 是用来指示参数域在封装对话框中位置的一个数。如@1 表示用户输入在第一个参数域，@2 表示用户输入在第二个参数域，等等。

对于上面这个例子，自然频率取决于参数 Natural Frequency 域、即新对话框中第一个参数域中键入的值，变量名(wn)和 Fransfer Fcn 模块的对话框中所用的名字相同。要为自然频率定义初始化命令，可用下面的命令：

wn = @1;

同样,阻尼系数是在参数 Damping Coefficient(第二个)域中键入的值,定义这两个参数的初始化命令为:

wn = @1;zeta = @2;

2. 初始化变量的作用范围

在参数 Initialization Commands 域中定义的变量对对话框来说是局部的,即它们只对对话框中的其它初始化命令和被封装模块的初始化命令来说,是可以使用的。另外,尽管被封装模块能够获得工作空间里的变量,但是初始化命令却不能获得工作空间里的变量。

例如,如果定义一个初始化命令为 wn=x * @1,其中 x 是工作空间里的一个变量,那么 SIMULINK 就认为出错。另外,wn 只对封装后的模块和它的基本模块(即例子中的 Transfer Fcn 模块)的对话框来说,是可以使用的。

四、Drawing Commands 域

在 Drawing Commmands 域中定义一个命令可以为一个新模块定义一个图标。在这个域中可以输入文本、一个画图命令或者是传递函数的参数。如果输入了一个很长的字符串,可以使用\n 来表示一行的结束。在\n 的前后无需插入空格。

对于上面的例子,可以键入一个文本标签:

Second Order\nSystem

有关创建其它种类图标的详细描述,请参阅后面 8.3"为封装模块创建图标"这一节。

五、Help String 域

Help String 域可以用来为封装后的模块随意定义一个帮助入口。当用户用鼠标单击封装后模块的对话框中的 Help 按钮时,在 Help String 这个域中键入的帮助文本就在屏幕上显示出来。

当在帮助文本中需要另起一行时,可以在需要另起一行的前面加上\n,在\n 的前后无需插入空格。

8.2.2 创建封装模块的图标和对话框

当用户在各个域中输入完值以后,用鼠标单击 OK 按钮,SIMULINK 就创建图标和对话框。在对话框中包含有用户定义的标号域、初始化命令域和帮助字符串。对于上面这个模块封装后的图标和对话框如图 8-4 所示。

当用户为这些域输入完值,并用鼠标单击 OK 按钮以后,初始化命令就把这些值送给被封装的 Transfer Fcn 模块,并把其中的两个参数赋给自然频率(wn)和阻尼系数(zeta)这两个变量。

8.2.3 创建封装模块参数的默认值

正如前面对话框中所显示的那样,当经过封装后所得的模块其对话框被打开的时候,Natural Frequency 和 Damping Coefficient 这两个参数域是空的。如果用户每次打开该模块时想为其设置默认值,就必须在保存模块的时候就保存其默认值。要做到这一点,首先在这些域中输入值,然后用鼠标单击 OK 按钮,那么当你保存这个模型的时候,这些值就和模块一起保存起来。

8.2.4 封装含有封装模块的子系统

用户可以在 SIMULINK 中创建复杂的、多级的封装,并在多层封装中传递变量。要创建这样的封装,只要在各层中采用相同的变量名来封装包含封装模块或其它各种模块的子系统。

当用户在封装模块其对话框中输入值时,SIMULINK 就把那些参数传递给相应的模块。

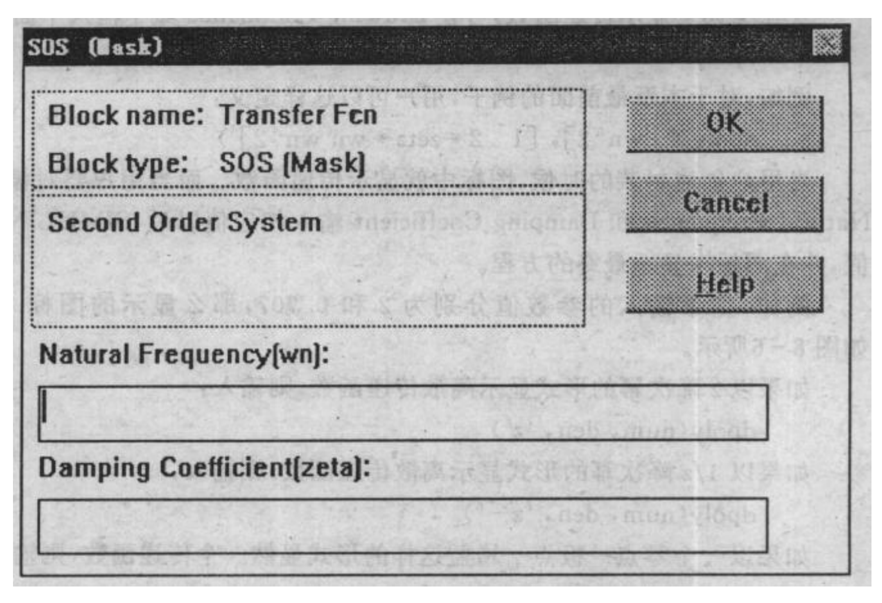


图 8-4

8.2.5 恢复一个封装模块

要恢复一个封装模块,首先要用鼠标选中该模块,然后从 Options 菜单中选择 Ummask 选项,SIMULINK 就用默认的图标方式显示该模块或子系统。

8.2.6 修改封装信息

要修改某个模块的封装信息,可以采用下面的步骤:

(1) 选中该模块。

(2) 在 Options 菜单中选择 MASK 命令,则显示当前的封装对话框。

(3) 修改封装信息,然后用鼠标单击 OK 按钮。

8.3 为封装模块创建图标

用户可以通过封装对话框的 Drawing Commands 域为封装模块定义一个更加具有特色的图标。采用这个域,可以为封装模块创建如描述性文本、状态方程或图形这样的图标。

8.3.1 在图标中显示文本

要在图标中显示文本,用户可在 Drawing Commands 域中输入文本即可。如果需要在文本中另起一行,只需要在其前面加上\n,而在\n 的前后无须插入空格。例如,图 8-5 是两个在 Drawing Commands 域中输入的两个文本注释和它们最终的图标。

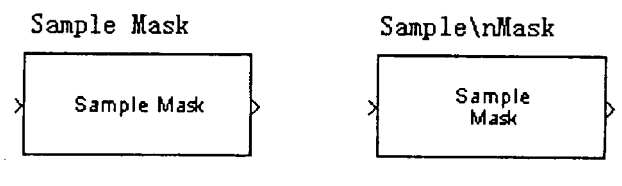


图 8-5

8.3.2 在图标中传递函数

要在图标中显示传递函数，可在 Drawing Commands 域中输入下面这条命令：

```
dpoly ( num, den )
```

例如，对于本章最前面的例子，用户可以这样定义：

```
dpoly ( [wn^2], [1  2 * zeta * wn wn^2] )
```

当用户创建封装的时候，图标中就显示传递函数。而当用户要对模型进行仿真并给参数 Natural Frequency 和 Damping Coefficient 输入相应值以后，SIMULINK 就计算传递函数的值，并在图标中显示最终的方程。

例如，如果输入的参数值分别为 2 和 0.707，那么显示的图标如图 8－6所示。

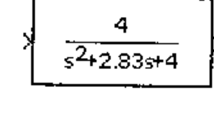


图 8－6

如果以 z 降次幂的形式显示离散传递函数，则输入：

```
dpoly(num, den, 'z')
```

如果以 1/z 降次幂的形式显示离散传递函数，则输入：

```
dpoly(num, den, 'z—')
```

如果以一个零点—极点—增益这样的形式显然一个传递函数，则输入：

```
droots(z, p, k)
```

当用户输入完上面任何一条命令并用鼠标单击 OK 按钮时，SIMULINK 就显示一个出错信息，并指出这些参数没有定义。只有当这些参数值在封装对话框中输入以后，SIMULINK 才能产生传递函数方程。

8.3.3 在图标中显示图形

在 Drawing Commands 域中输入 plot 命令，可以使用户在封装模块的图标中显示图形。除非数据非常复杂，否则的话，这里的 plot 命令和 MATLAB 的 plot 命令非常相似。

对于 x、y 的数据没有任何限制，而且画图时自动调整比例。例如，下面的命令将使 Source library 中 Pulse Generator 模块在其图标中显示图形。

```
plot (0,0,100,100,[90,75,75,60,60,35,35,20,20,10],[20,20,80,80,20,20,80,80,
20,20])
```

要检查所使用的画图命令，并在封装模块的图标上显示图形，可采用下面的步骤：

(1) 把模块拷贝到用户的模型窗口。

(2) 选中模块，然后在 Option 菜单中选择 Mask 命令。SIMULINK 就显示封装对话框，检查 Drawing Commands 域。

(3) 要获得一个封装模块，在鼠标上单击 OK 按钮。

如果画图取决于变量，那么在为这些变量输入值以前，图标将显示三个问号。

8.3.4 用图标编辑功能创建图标

MATLAB 中的 iconedit 为用户提供了一个简单的方法来创建一个简单的图形图标。要运行 iconedit，可在 MATLAB 命令窗口键入 iconedit 即可。MATLAB 首先要求用户辨认出模型窗口和需要创建图标的模块，然后用一个轴和网格线来显示一个图形窗口。

要进行画图，可以采用下面这些简单的步骤：

(1) 把一个十字形鼠标指针定位到你需要画图的起始位置，并在鼠标按钮上单击一下，那么在鼠标指针处网格上显示一个星号。如果在一个错误的位置上按了鼠标，可键入 d 把该点

删除。

(2) 把鼠标指针移到下一个需要显示的位置,并在鼠标按钮上单击一下,iconedit 就把两点用连线连接起来,并在第二个点处画一个星号。

(3) 连续移动鼠标指针和单击鼠标按钮,重复上面的步骤。一旦需要终止绘图,可键入 q。iconedit 就在命令窗口上显示生成这个图所需要的命令,然后在屏蔽对话框的 Display commands 域中键入这些命令,就可生成所需要的图标。

第九章　SIMULINK 模块的索引

本章给出了 SIMULINK 中所有模块的一些参考信息。这些模块是以字母表的顺序排列的。主要包括以下一些信息:

(1) 模块名及其相应的图标。

(2) 模块的用途。

(3) 模块所在的库名。

(4) 模块的使用说明。

(5) 模块的对话框及其需要用户输入的参数(如果这种模块有这样要求的话)。

(6) 模块的特性。

9.1　模块的特性

模块的特性包括:

(1) 标量扩展,即是否对标量值扩展为向量值。有些模块把标量输入或参数扩展为相应维数的向量,有关其详细描述请参见第六章中的内容。

(2) 采样时间,即模块的采样时间是由自已决定,还是由驱动模块来决定。

(3) 状态个数,即有多少连续和离散状态的各抒己见。

(4) 直接前馈,即模块是否直接向前传输,有关其详细描述请参见第七章中的内容。

9.2　模块的分类及其用途

在 SIMULINK 中,各个模块按照其不同的功能和用途分成了几个库,这些库包括:

(1) Source library。这个库是由产生源信号和源数据的模块组成的,表 9-1 列出了这个库中的所有模块。

表　9-1

模块名	用　途
Band-Limited White Noise	把白噪声加到连续系统中
Chip Signal	产生一个频率不断增大的正弦波
Clock	显示和提供仿真时间
Constant	产生一个常值
Digital Clock	在规定的采样间隔产生仿真时间
From File	从文件中读数据
From Workspace	从工作面上定义的矩阵中读数据
Pulse Generator	在固定的时间间隔产生脉冲
Random Number	产生正态分布的随机数

续 表

模块名	用 途
Repeating Sequence	产生规律重复的任意信号
Signal Generator	产生各种不同的波形
Sine Wave	产生一个正弦波
Step Input	产生一个阶跃函数

(2) Sinks library。这个库是由用于显示和输出的模块组成的，表 9-2 列出了这个库中的所有模块。

表 9-2

模块名	用 途
Auto—Scale Graph Scope	在 Matlab 自动调整显示比例的图形窗口显示信号
Graph Scope	在 Matlab 图形窗口显示信号
Hit Crossing	在规定值附近增加仿真步数
Scope	在仿真过程中显示信号
Stop Simulation	当输入不为零时停止仿真
To File	把数据输出到文件中
To Workspace	把数据输出到工作面上定义的一个矩阵中
XY Graph Scope	在 Matlab 图形窗口显示信号的 X—Y 图

(3) Discrete library。这个库是由描述离散时间系统的模块组成的。表 9-3 列出了这个库中的所有模块。

表 9-3

模块名	用 途
Discrete—Time Integrator	对一个信号进行离散积分
Discrete—Time Limited Integrator	对一个信号进行离散有限积分
Discrete State—Space	建立一个离散状态空间模型
Discrete Transfer Fcn	建立一个离散传递函数
Discrete Zero—Pole	以零极点形式建立一个离散传递函数
Filter	建立 IIR 和 FIR 滤波器
First—Order Hold	建立一阶采样保持器
Unit Delay	对一个信号延迟一个采样周期
Zero—Order Hold	建立一个采样周期的零阶保持器

(4) Linear library。这个库由标准线性函数和线性系统的模块组成。表 9－4 列出了这个库中的所有模块。

表 9－4

模块名	用途
Derivative	对输入信号进行微分
Gain	对输入信号乘上一个常数增益
Inner Product	对输入信号进行点积
Integrator	对输入信号进行积分
Matrix Gain	对输入信号乘上一个矩阵增益
Slider Gain	以滑动形式改变增益
State - Space	建立一个线性状态空间模型
Sum	对输入信号进行求和
Transfer Fcn	建立一个线性传递函数
Zero - Pole	以零极点形式建立一个传递函数

(5) Nonlinear library。这个库由描述非线性函数的模块组成。表 9－5 列出了这个库中的所有模块。

表 9－5

模块名	用途
Abs	输出输入信号的绝对值
Backlash	用放映的方式模仿一个系统的特性
Combinatorial	建立一张真值表
Coulombic Friction	在原点不连续而在原点以外具有线性增益
Dead Zone	提供一个死区
Fcn	对输入进行规定的表示
Limited Integrator	在规定的范围内进行积分
Logical Operator	对输入进行规定的逻辑运算
Look - up Table	对输入进行分段的线性映射
MATLAB Fcn	定义一个函数对输入信号进行处理
Memory	输出本模块上一步的输入值
Product	对输入进行乘积运算
Quantizer	对输入进行量化处理
Rate Limiter	限制信号的变化速率

续 表

模块名	用　　途
Relational Operator	对输入进行一定的关系运算
Relay	在两个值中轮流输出
Reset Integrator	在仿真中对积分器进行重新初始化
Saturation	对输入信号进行限幅
Sign	符号函数
Switch	在两个输入之间进行开关
Transport Delay	对输入信号进行一定的延迟
2-D Look-Up Table	对两个输入进行分段的线性映射
Variable Transport Delay	对输入信号进行不定量的延迟

(6) Connections library。这个库由一些实现模块与模块之间连接功能的模块组成的。表9-6列出了这个库中的所有模块。

表 9-6

模块名	用　　途
Demux	把向量信号分开输出
Inport	给系统提供一个外部输入
Mux	把几个信号合并成相量形式
Outport	给系统规定一个输出
Subsystem	表示一个系统在另外一个系统中

9.3 模块的图标及使用说明

下面对这些模块按照其英文字母顺序分别予以介绍：

1. Abs

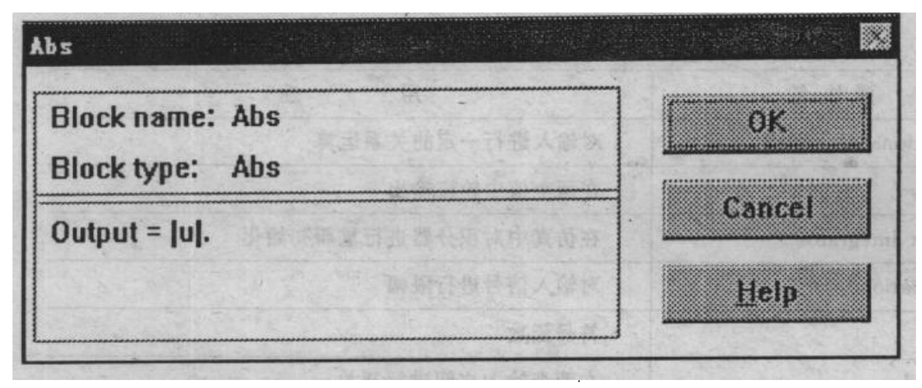
图　标



所在库名　nonlinear library

说　明　Abs 模块的功能是输出输入信号的绝对值。这个模块接收一个输入并产生一个输出，输入和输出既可以是标量，也可以是向量。

对话框



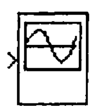
特　　性　　采样时间　　从它的驱动模块中继承

状态个数　　0

直接输出　　是

2. Auto－Scale Graph Scope

图　　标



所在库名　　Sinks library

说　　明　　Auto－Scale Graph Scope 模块的功能是把它的输入信号在 MATLAB 的图形窗口中显示出来，并自动调整画图比例，以便使数据能在屏幕上显示出来。本模块接收一个输入，这个输入既可以是标量，也可以是向量，并以仿真时间为 x 轴，输入数据为 y 轴在屏幕上显示出来。最初，显示区域在 x 方向由参数 Initial Time Range 规定，在 y 方向由 Initial y－min 和 Initial y－max 规定。在仿真开始后，y 值根据信号的最大和最小值自动调整比例显示出来。

本模块把输入和时间数据存到一个缓冲器中，缓冲器的大小由参数 Storage points 确定，这个参数是本模块所能显示的最大数据点个数。

如果本模块的输入是一个向量，则各个量将以参数 Line type 规定的值，以不同的颜色、线型和打印符号显示出来。

有关如何使用本模块的方法，SIMULINK 提供了一个演示程序，你可在命令窗口中键入 lorenz2 即可。

参　　数　　Initial Time Range

x 轴（时间轴）初始长度。默认值为 5 s。

Initial y－min

y 轴初始下界。默认值为－10。

Initial y－max

y 轴初始上界。默认值为 10。

Storage points

所能显示的最大点数。默认值为 200 个点。

Line type

用来输出的颜色、线型、符号。每个线型之间用"/"隔开。有关这些代码的内容，请参见 MATLAB 中 Plot 命令。默认值为‘y -/g —/c—. /w:/m * . ro/b+’。

对 话 框

Storage scope. (Mask)

Block name: Auto-Scale Graph
Block type: Storage scope. (Mask)

Storage scope using MATLAB graph window.
Enter plotting ranges and line type.

OK
Cancel
Help

Initial Time Range:
5

Initial y-min:
-10

Initial y-max:
10

Storage pts.:
200

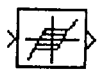
Line type (rgbw-.:xo):
'y-/g--/c-./w:/m*/ro/b+'

特　　性　　　采样时间　　　从它的驱动模块中继承

　　　　　　　状态个数　　　9 个离散的状态(内部使用)

3. Backlash

图　　标



所在库名　　Nonlinear library

说　　明　　Backlash 模块的功能是在输出不变区以外，当输入中发生多大的变化时，在输出中发生同样大小的变化；而在输出不变区中，不管输入变化否，输出都不发生变化。

本模块接收单个输入，这个输入既可以是标量，也可以是向量。

在开始仿真时，输出的初始位置由参数 Initial output value 确定，输出不变区的宽度由参数 Dead width 确定。初始的输入值在输出不变区的中心。本模块的输出由下列条件确定：

(1) 当输入在输出不变区的范围内，输出保持常值。

(2) 当输入到了输出不变区两端的任一端，输入在哪一方向上的变化将导致输出在相同方向发生同样的变化。

参　　数　Deadband width

输出不变区的宽度。

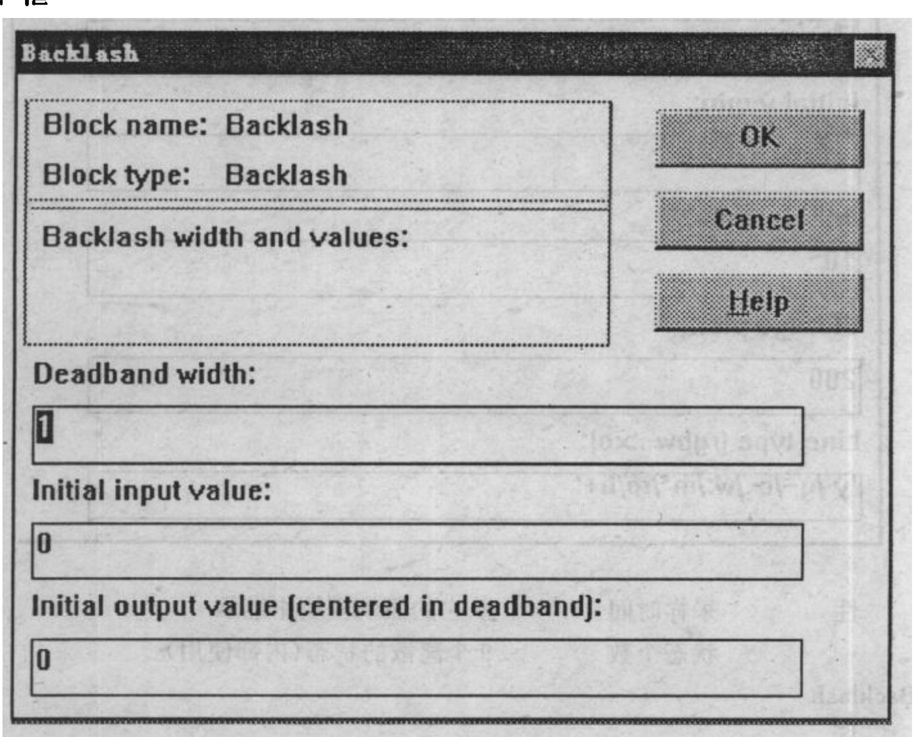
Initial input value

初始输入值和初始输出值之间的差不能超过输出不变区的宽度。

Initial output value

初始输出值。

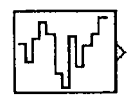
对 话 框



特　　性	标量扩展	参数
	采样时间	从它的驱动模块中继承
	状态个数	0
	直接输出	是

4. Band - Limited White Noise

图　　标



所在库名　　Sources library

说　　明　　Band－Limited White Noise 模块的功能为连续系统产生白噪声输入。

本模块产生一个输出，它既可以是标量，也可以是向量。

本模块产生的白噪声是连续白噪声的近似，它是将白噪声通过一个 Zero－Order Hold 模块实现的。当你增加采样时间的时候，为了防止能量损失，本模块将加大信号幅值。

参　　数　　Noise Power

Noise Power 和 Seed 可以是具有相同长度的向量，用来产生一个白噪声向量。默认值为[0.1]。

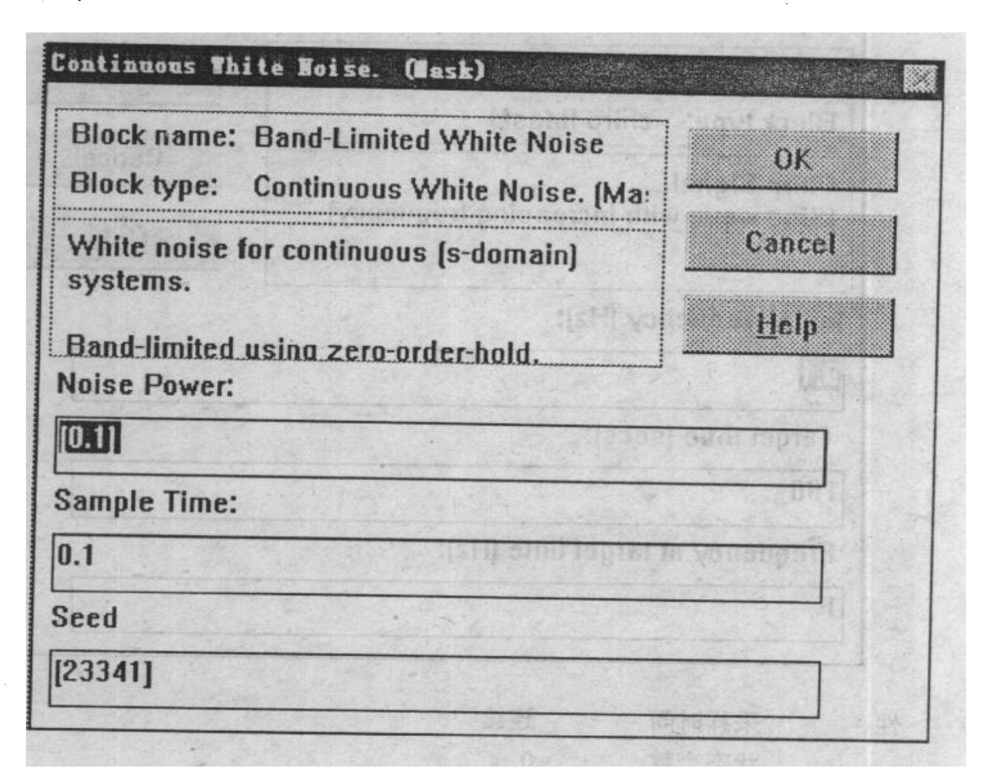
Sample Time

为了使仿真进行的快一些，在与系统动态特性相一致的前提下，尽可能把采样时间设得大一点。Sample Time 必须是一个标量。

Seed

Seed 是用来产生白噪声，Noise Power 和 Seed 可以是具有相同长度的向量来产生一个白噪声向量。默认值为[23341]。

对 话 框



特　　性　　采样时间　　离散

　　　　　　状态个数　　0

5. Chirp Signal

图　　标



所在库名　Sources library

说　　明　Chirp Signal 模块的功能是产生一个频率随时间线性增长的正弦波。本模块可用于非线性系统的谱分析，它产生一个标量输出。

参　　数　Initial frequency(Hz)

所产生信号的初始频率。默认值为的 0.1 Hz。

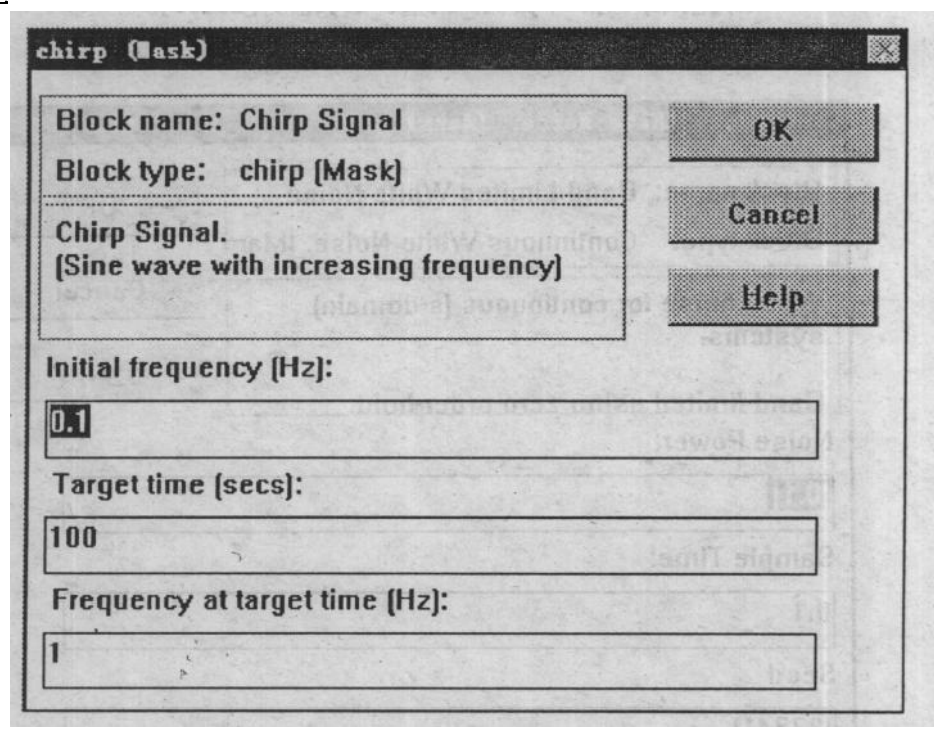
Target time(secs)

由 Initial frequency 增加到 Frequency at target time 所需的时间。在这个时间里面，频率将以同样的速率不断递增。默认值为 100 s。

Frequency at target time

最终频率。默认值为 1 Hz。

对 话 框



特　　性	采样时间	连续
	状态个数	0

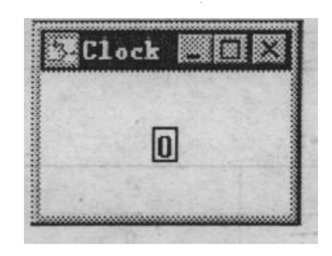
6. Clock

图　　标

所在库名　　Sources library

说　　明　　Clock 模块的功能是在每一个仿真步长内输出目前的仿真时间。当本模块打开时，时间就显示在窗口中。如果在仿真时打开本模块就会使仿真速度减慢。本模块对一些需要仿真时间的其它模块来说是有用的。

对 话 框



特　　性　　采样时间　　连续

　　　　　　状态个数　　0

7. Combinatorial Logic

图　　标

所在库名　　Nonlinear library

说　　明　　Combinatorial Logic 模块的功能是为可编程逻辑器件、逻辑电路、决策表和其它的布尔型表达式建立一张标准的真值表。用户可以用本模块和 Memory 模块来实现有限状态机或触发器。

用户可以用一个矩阵的参数作为真值表的参数，矩阵的每一列对应于不同的输出，每一行包含输入的不同组合。行的总数为 2^n（n 为输入的个数）。

本模块的输入为一个向量，并根据这个向量从真值表中输出相应的值。真值表的每一行按照传统的习惯进行排列，第一行为[0 0 0...0]（所有的输入为零），最后一行为[1 1 1...1]（所有的输入为 1），其中数的个数由输入的个数来决定。所有的非零输入值都看作 1。

第一个输出值对应于输入向量[0 0 0...0]，第二个输出值对应于输入向量[0 0...0 1]，第三个输出值对应于输入向量[0 0...1 0]，以此类推。

本模块有一个输入端口，一个输出端口。

例 1　两个输入的与运算

例如，两个输入的与运算对应的输出为：

[0;0;0;1]

这对应于真值表 9－7。

如果输入向量为[1,0]，即输入向量对应于真值表的第三行。因此，对应的输出为 0。

例 2　一个二进制加法器

这个电路有三个输入，加位(a)、被加位(b)和进位位(c)。它有两个输出，一个是和位(s)，

另一个是进位位(c′)。这个电路的真值表如表 9-8 所示。

表 9-7

输入 1	输入 2	输　出
0	0	0
0	1	0
1	0	0
1	1	1

表 9-8

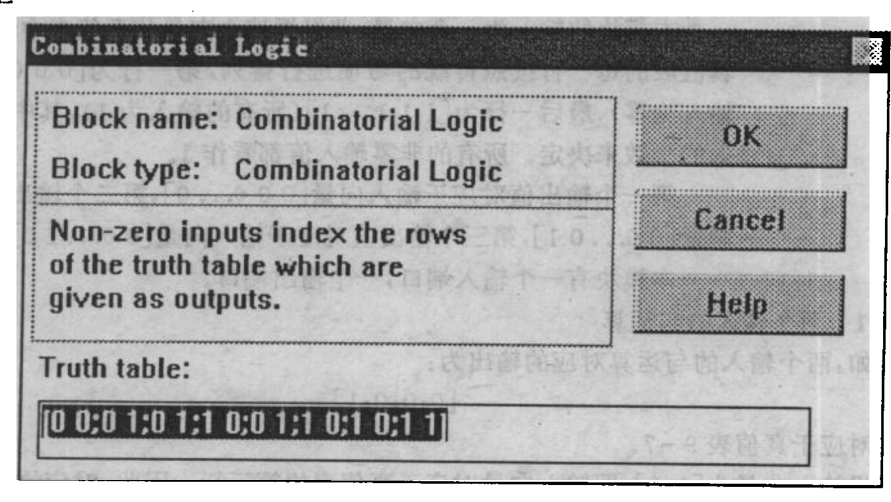
a	b	c	c′	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

用 Combinatorial Logic 模块来实现这个加法器,用户可以输入一个由 c′和 s 的真值表参数组成的 8 * 2 的矩阵即可。

参　　数　　　Truth table

输出矩阵。每一列对应一个变量输出,每一行对应于真值表的每一行。

对 话 框



特　　性　　　输入向量的宽度　　　逻辑表的长度取 2 的对数。

输出向量的宽度　　逻辑表列的个数。

采样时间　　从它的驱动模块中继承

状态个数　　0

直接输出　　是

8. Constant

图　　标　　1

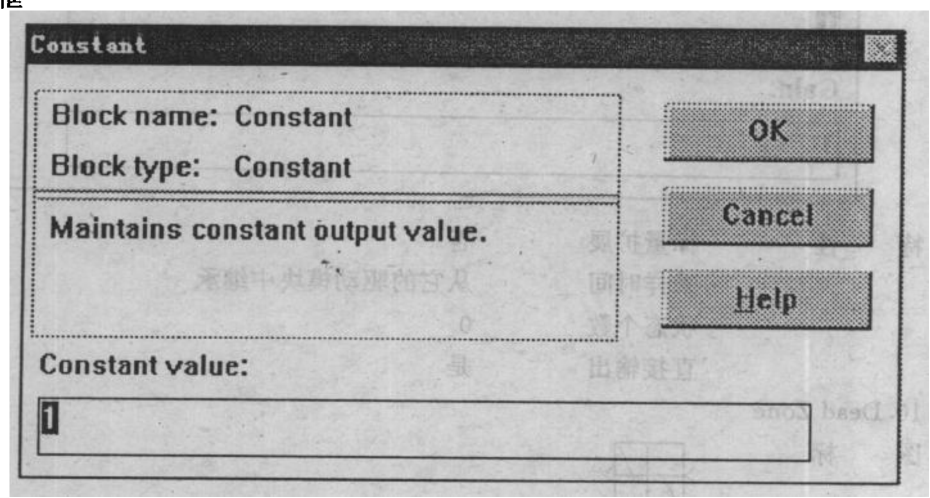
所在库名　　Sources library

说　　明　　Constant 模块的功能是产生一个常值。本模块产生一个输出，它既可以是标量，也可以是向量，这主要取决于参数 Constant value。由参数 Constant value 规定的常值在图标上显示出来。

参　　数　　Constant value

本模块的输出值。如果定义了一个向量，则输出的值就是一个常值向量。默认值为 1。

对话框

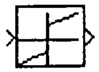


特　　性　　采样时间　　固定

状态个数　　0

9. Coulombic Friction

图　　标



所在库名　　Nonlinear library

说　　明　　Coulombic Friction 模块是在零点不连续，而在其它点上具有线性增益的模块。它是由下面的函数来实现的：

$$y=sign(u)*(Gain*abs(u)+Offset)$$

其中：y 是输出，u 是输入，Gain 和 Offset 是模块中定义的参数。

本模块接收一个输入，并产生一个输出。两者既可以是标量，也可以是

向量。

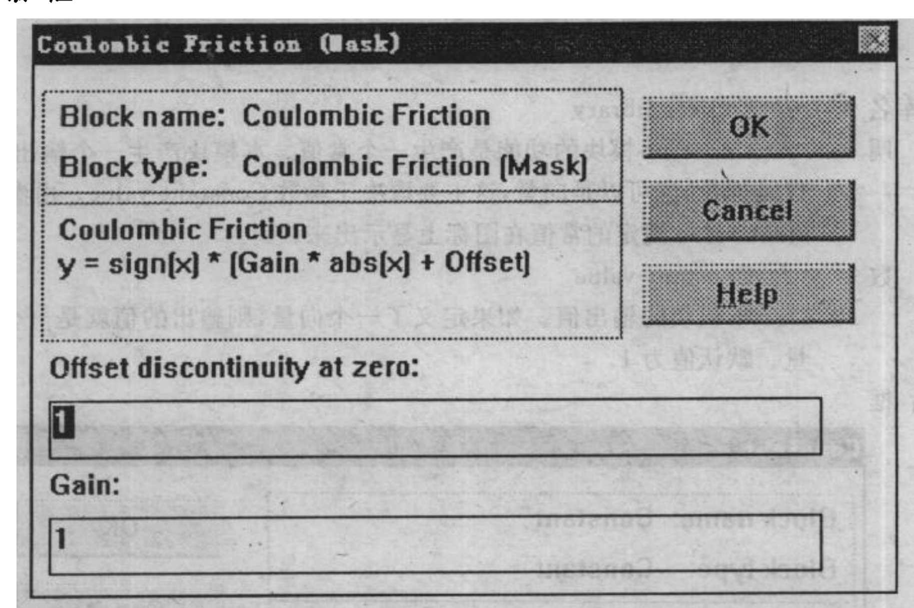
参　　数　　Offset discontinuity at zero

对所有输入值的偏移量。

Gain

在非 0 输入点的信号增益。

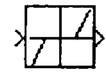
对 话 框



特　　性	标量扩展	否
	采样时间	从它的驱动模块中继承
	状态个数	0
	直接输出	是

10. Dead Zone

图　　标



所在库名　　Nonlinear library

说　　明　　Dead Zone 模块的功能是在规定的死区范围内模块没有输出。死区的上界和下界分别由参数 Start of dead zone 和 End of dead zone 来确定。本模块的输出取决于输入与死区之间的关系：

(1) 如果输入大于死区的下界而小于上界，则输出为 0。

(2) 如果输入大于上界，则输出等于上界减去输入。

(3) 如果输入小于下界，则输出等于输入减去下界。

(4) 如果下界和上界相等，则输出值等于输入减去死区值。

本模块接收一个输入并产生一个输出，两者既可以是标量，也可以是向量。

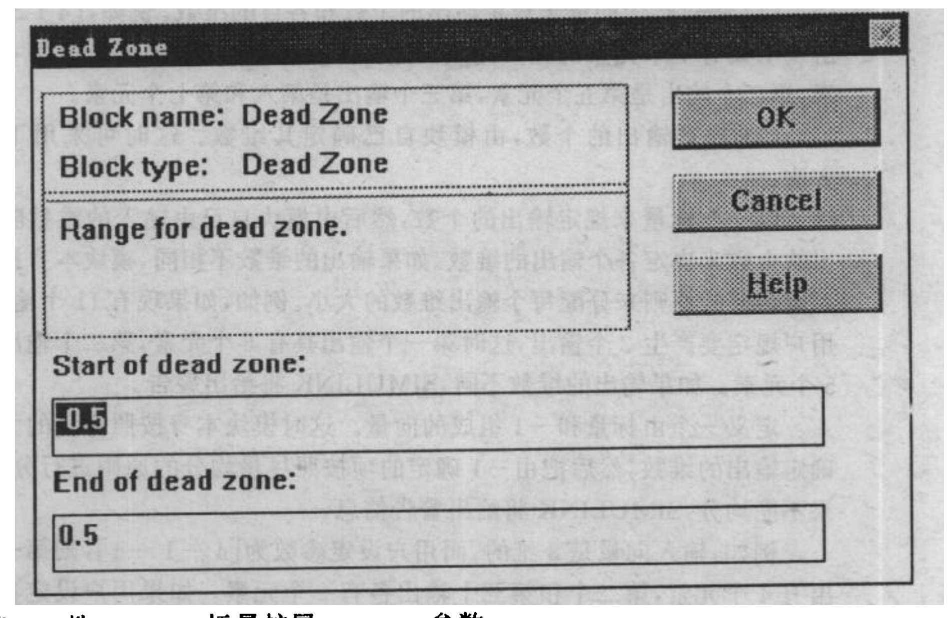
参　　数　　Start of dead zone

死区的下界。

End of dead zone

死区的上界。

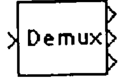
对 话 框



特 性	标量扩展	参数
	采样时间	从它的驱动模块中继承
	状态个数	0
	直接输出	是

11. Demux

图 标



所在库名 Connections library

说 明 Demux 模块的功能是把一个向量输入分成相应个数的标量进行输出。

Demux 模块的输入可以是一个具有任意维数的向量，而输出可以有指定个数，每一个输出既可以是向量，也可以是标量。SIMULINK 能根据本模块输出的实际个数在图标上显示出来。

参 数 Number of outputs

输出的个数和维数。注意：输出维数的总数必须和输入维数匹配。

对于标量输出，参数 Number of outputs 可设为输出的个数或－1。本模块就为输入向量中的每个元素建立一个对应的输出。第 m 个输出就是输入向量的第 m 个元素。

对于多个具有相同维数的向量输出，参数可设为输出的个数。例如，如输入向量有 12 个元素，而用户又想产生 4 个具有 3 个元素的输出向量，参

数可设为 4。

对于多个具有不同维数的向量输出，用户可以规定输出的个数和各自的维数；或者由用户规定输出的个数，再由模块自己确定其维数。下面介绍这两种办法：

(1) 通过一个向量来规定输出的个数和各自的维数。例如，[4 1 2]表示由一个具有 7 个元素的输入向量产生 3 个输出，第一个输出是前面 4 个元素，第二个输出是第五个元素，第三个输出是第六和第七个元素。

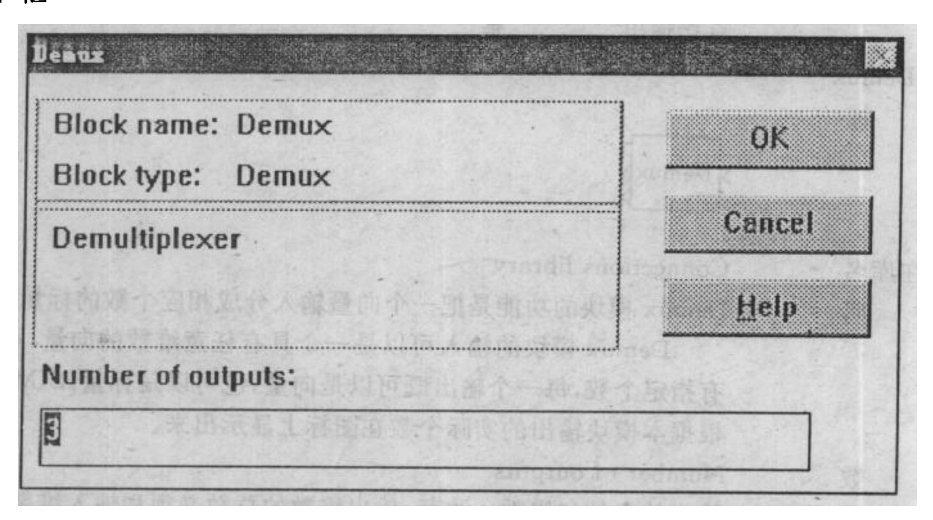
(2) 规定输出的个数，由模块自己确定其维数。这时可采用下面的方法：

以一个标量来规定输出的个数，然后由模块自己由输入的维数除以输出的个数来决定各个输出的维数。如果输出的维数不相同，模块本身按照尽可能相同的原则来分配每个输出维数的大小。例如，如果现有 11 个输入，而用户规定要产生 2 个输出，这时第一个输出具有 6 个元素，第二个输出具有 5 个元素。如果输出的维数不同，SIMULINK 将给出警告。

定义一个由标量和 −1 组成的向量。这时模块本身按照标量的大小来确定输出的维数，然后把由 −1 确定的项按照尽量均分的原则进行分配，如果不能均分，SIMULINK 将给出警告信息。

例如：输入向量是 8 维的，而用户设定参数为[4 −1 −1]，则第一个输出有 4 个元素，第二个和第三个输出各有 2 个元素。如果用户设定参数为[5 −1 −1]，则第一个输出有 5 个元素，第二个输出有 2 个元素，第三个输出有 1 个元素，同时，SIMULINK 将给出警告信息。

对话框



特　性	标量扩展	N/A
	采样时间	从它的驱动模块中继承
	状态个数	0
	直接输出	是

12. Derivative

图　　标

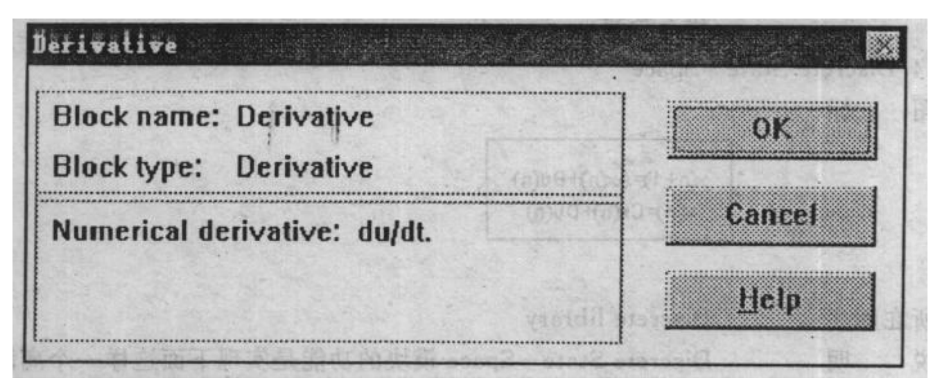
du/dt

所在库名　　Linear library

说　　明　　Derivative 模块通过计算$\frac{\Delta u}{\Delta t}$来近似输入的微分，其中：$\Delta u$ 是输入的变化量，Δt 是时间的变化量。本模块接收一个输入并产生一个输出，两者既可以是标量，也可以是向量。模块的初始输入为零。

结果的准确性取决于仿真步长。仿真步长越小，输出越精确越平滑。用 linmod 来线性化带有微分模块的系统是非常麻烦的，有关如何避免这些问题的方法，请参见第七章中相应内容。

对 话 框



特　　性

标量扩展	N/A
采样时间	从它的驱动模块中继承
状态个数	0
直接输出	是

13. Digital Clock

图　　标

12:34

所在库名　　Sources library

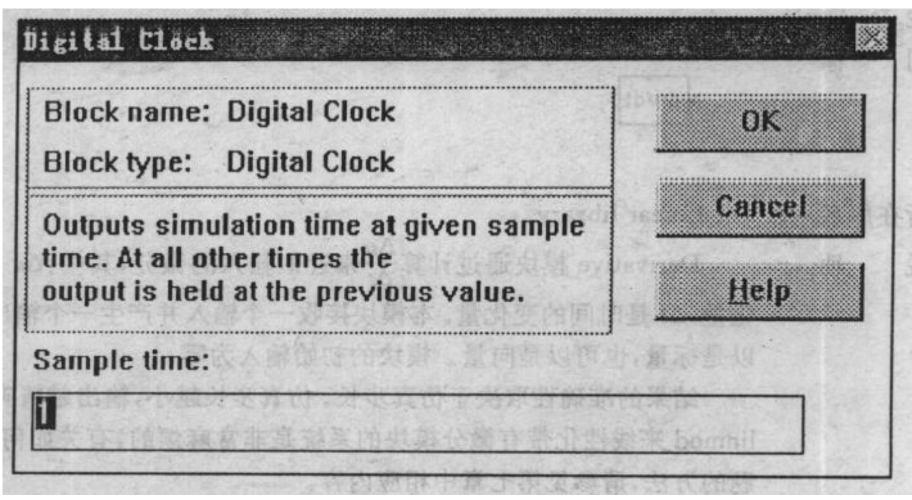
说　　明　　Digital Clock 模块的功能是在规定的采样间隔内输出当前的仿真时间，而在其它时间间隔内，输出一直保持前一步值。

当用户在离散系统中需要知道当前的时间时，则采用本模块而不采用 Clock 模块（输出连续时间），采用本模块可以提高多个采样速率系统所生成代码的效率。

参　　数　　Sample time

采样间隔。默认值是 1 s

对 话 框



特　　性	采样时间	离散
	状态个数	0

14. Discrete State - Space

图　　标

x(n+1)=Ax(n)+Bu(n)
y(n)=Cx(n)+Du(n)

所在库名　　Discrete library

说　　明　　Discrete State－Space 模块的功能是实现下面这样一个离散系统

$$x(n+1)=Ax(n)+Bu(n)$$

$$y(n)=Cx(n)+Du(n)$$

其中：u 是输入，x 是状态，y 是输出。

本模块接收一个输入，并产生一个输出。两者既可以是标量，也可以是向量。输入向量的维数取决于矩阵 B 和 D 的维数，输出向量的维数取决于矩阵 C 和 D 的维数。

参　　数　　A,B,C,D

上面方程的系数矩阵。

A 必须是 n * n 的矩阵，其中 n 是状态的个数。

B 必须是 n * m 的矩阵，其中 m 是输入的个数。

C 必须是 q * n 的矩阵，其中 q 是输出的个数。

D 必须是 q * m 的矩阵。

Initial conditions

初始状态向量。如果没有输入，则认为是零。

Sample time

采样周期。Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许

在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

对 话 框

Block name: Discrete State-Space
Block type: Discrete State-Space

Discrete state-space model matrices:
x[n+1] = Ax[n]+Bu[n]
y[n] = Cx[n]+Du[n]

OK
Cancel
Help

A:
1

B:
1

C:
1

D:
1

Initial conditions:

Sample time:
1

特　　性	采样时间	离散
	状态个数	可变，取决于矩阵 A 的维数
	直接输出	只要 D 不为零

15. Discrete - Time Integrator

图　　标

$\frac{1}{z-1}$

所在库名　　Discrete library

说　　明　　Discrete－Time Integrator 模块的功能是实现离散的欧拉积分，其离散传递函数为

$$y=\frac{T_s}{z-1}u$$

其中：y 是输出，T_s 是采样周期，u 是输入。

本模块在构造纯粹的离散系统时可用来代替连续的积分器（1/S）模块。

本模块接收一个输入，并产生一个输出。两者既可以是标量，也可以是向量。

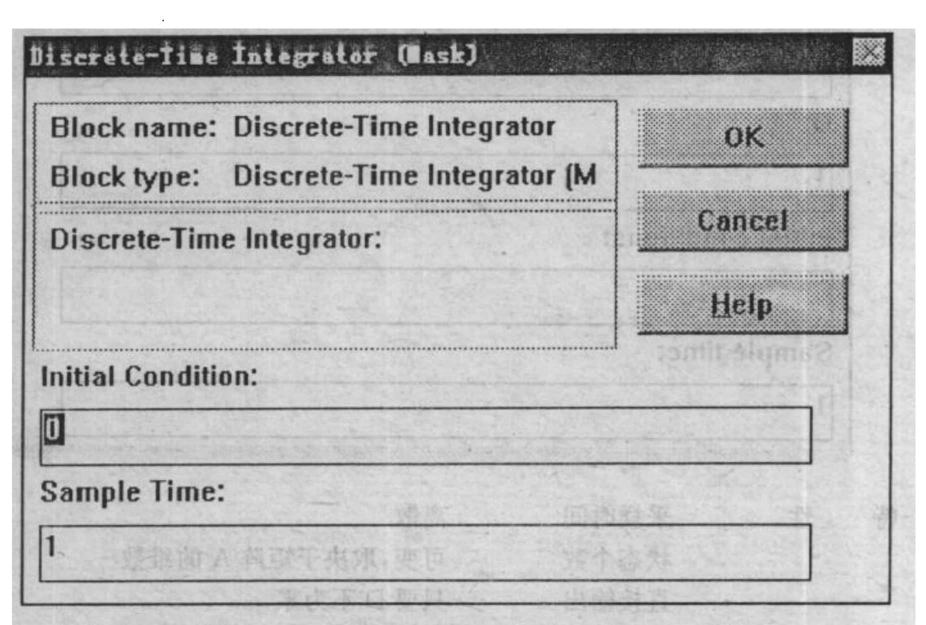
参　数　　Initial Condition

初始条件既可以是一个向量，也可以是一个标量。如果输入是一个向量，初始条件也是个向量，那么它们的维数应一致。如果初始条件是一个标量，则进行标量扩展。输入向量中第 m 个元素就以初始值向量中的第 m 个元素为初始条件进行积分，生成输出向量中的第 m 个元素。

Sample Time

采样周期。Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

对话框

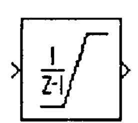


特　性

标量扩展	参数 Initial Condition
采样时间	离散
状态个数	可变
直接输出	不

16. Discrete－Time Limited Integrator

图　标



所在库名　Discrete library

说　　明　Discrete－Time Integrator 模块的功能是实现离散的欧拉积分，并限制积分超过规定的值。本模块接收一个输入，并产生一个输出。两者既可以是标量，也可以是向量。

参　　数　Lower bound

输出的下界。

Upper bound

输出的上界。

Initial Condition

初始条件，它既可以是一个向量，也可以是一个标量。如果输入是一个向量，初始条件也是个向量，那么它们的维数应一致。如果初始条件是一个标量，则进行标量扩展。输入向量中第 m 个元素就以初始值向量中的第 m 个元素为初始条件进行积分，生成输出向量中的第 m 个元素。

Sample Time

采样周期，Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

对 话 框

Discrete-Time Limited Integrator. (Mask)

Block name: Discrete-Time Limited Integrator

Block type: Discrete-Time Limited Integrator. (M

Discrete-Time Limited Integrator.

OK

Cancel

Help

Lower bound:

-1

Upper bound:

1

Initial condition:

0

Sampling Time:

1

特　　性	标量扩展	参数 Initial Condition 和 bound
	采样时间	离散
	状态个数	由驱动模块或参数中继承
	直接输出	否

17. Discrete Transfer Fcn

图　　标

$$\frac{1}{z+0.5}$$

所在库名　　Discrete library

说　　明　　Discrete Transfer Fcn 模块的功能是实现一个具有下列形式的离散传递函数：

$$H(z)=\frac{mum(z)}{den(z)}=\frac{mum(1)z^{nn-1}+num(2)z^{nn-2}+\cdots+num(nn)}{den(1)z^{nd-1}+den(2)z^{nd-2}+\cdots+den(nd)}$$

其中 nn 和 nd 分别是分子和分母的阶次。行向量 num 和 den 包含有分子和分母两个多项式的系数，并以 z 的递减次幂的形式排列。一个具有 n 个元素的向量定义了一个 n－1 次的多项式。分母的阶次必须大于等于分子的阶次。本模块接受一个标量输入，并产生一个标量输出。

Discrete Transfer Fcn 模块在控制领域用来表示离散系统，而在信号处理领域用来表示滤波器。

根据 Discrete Transfer Fcn 模块对分子和分母多项式的定义，在其图标上显示不同的分子和分母多项式，有关其详细的信息请参见 Transfer Fcn 模块。

参　　数　　Numerator

分子多项式系数向量。默认值为[1]。

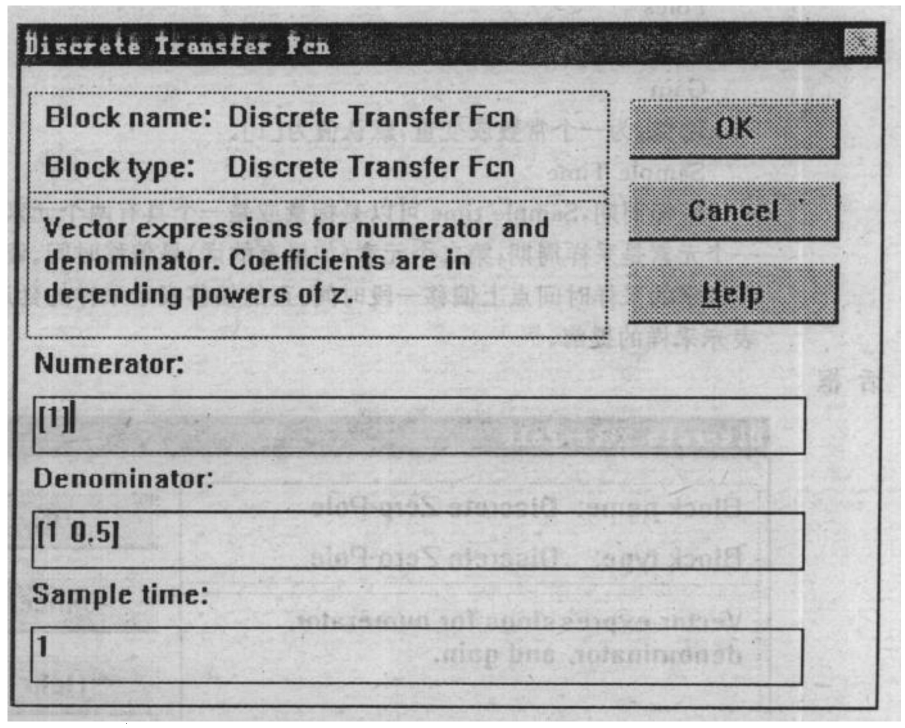
Denominator

分母多项式系数向量。默认值为[1 0.5]。

Sample Time

采样周期，Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

对 话 框



特　　性	标量扩展	N/A
	采样时间	离散
	状态个数	其状态个数为分母的阶次
	直接输出	如果分子和分母阶次相同，则直接输出

18. Discrete Zero－Pole

图　　标

$$\frac{(z\text{-}1)}{z(z\text{-}0.5)}$$

所在库名　　Discrete library

说　　明　　Discrete Zero－Pole 模块的功能是建立一个具有指定零极点、以延迟算子 z 表示的离散系统。本模块接受一个标量输入，并产生一个标量输出。

对单输入单输出系统的传递函数可表示为分式形式或零点－极点－增益两种形式，即

$$H(z)=K\frac{Z(z)}{P(z)}=K\frac{(z-Z(1))(z-Z(2))\cdots(z-z(n))}{(z-P(1))(z-P(2))\cdots(z-P(n))}$$

其中 Z 表示零点向量，P 表示极点向量，K 表示标量增益。

根据 Discrete Zero－Pole 模块对其参数的定义不同，在其图标上显示方式就不同，有关其详细的信息请参见 Zero－Pole 模块。

参　　数　　Zeros

零点向量，默认值为[1]。

Poles

极点向量，默认值为[0;0.5]。

Gain

增益，为一个常数或变量，默认值为[1]。

Sample Time

采样周期，Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表示采样的延迟，负的偏移表示采样的提前。

对 话 框

Discrete Zero-Pole

Block name: Discrete Zero-Pole

Block type: Discrete Zero-Pole

Vector expressions for numerator, denominator, and gain.

OK

Cancel

Help

Zeros:

[1]

Poles:

[0; 0.5]

Gain:

[1]

Sample time:

1

特　　性

标量扩展	不
采样时间	离散
状态个数	极点向量的长度
直接输出	如果极点数和零点数相同，则直接输出

19. Fcn

图　　标

sin(u[1])

所在库名　Nonlinear library

说　　明　Fcn 模块的功能是对输入进行一些符合 C 语言规范的数学表达式处理。这些表达式由一个或多个下列这些成分组成的：

(1) u，是模块的输入。如果 u 为向量，则 u[i]表示向量的第 i 个元素；u[1]或单独 u 表示第一个元素。

(2) 常数。

(3) 数学算子(+、-、*、/)

(4) 关系算子(==、!=、><、>=、<=)。如果关系成立，则表达式返回 1；否则，返回零。

(5) 逻辑算子(&&、||、!)。如果关系成立，则表达式返回 1；否则，返回零。

(6) 括弧。

(7) MATLAB 函数，包括：abs，acos，asin，atan，atan2，ceil，cos，cosh，exp，fabs，floor，hypot，ln，log，log10，pow，power，rem，sgn，sin，sinh，sqrt，tan，tanh。

(8) 工作空间上的变量。对矩阵或向量中元素必须明确指出，如用 A(1,1)而不用 A 来表示矩阵的第一个元素。

本模块的输入既可以是标量，也可以是向量。但输出总是标量。

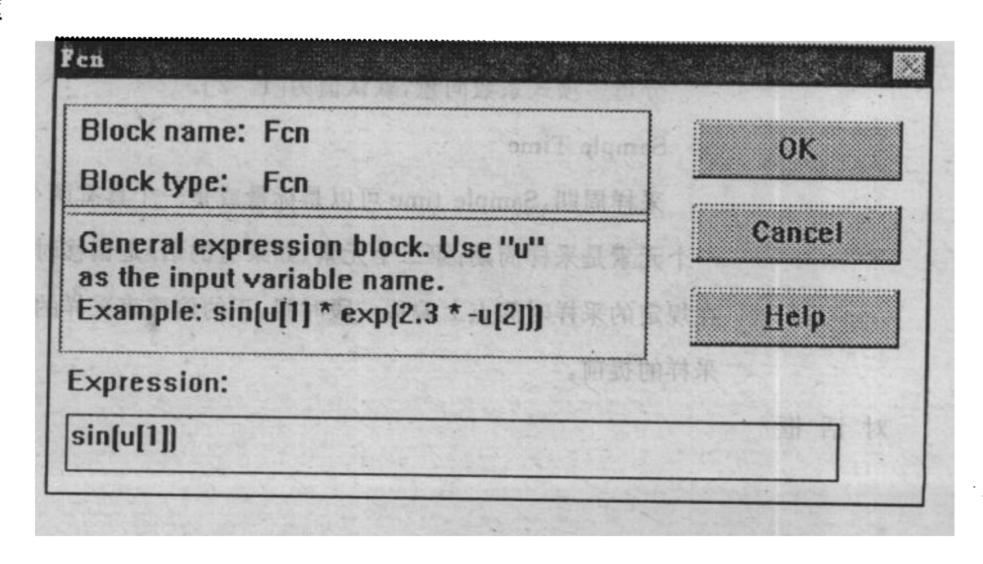
参　　数　Expression

必须是符合 C 语言规范的表达式，并且由上面提到的 8 部分来组成。这里的表达式与 MATLAB 表达式的主要区别在于：

(1) 表达式不能完成矩阵运算。

(2) 表达式输入变量中的元素用方括号进行索引，而工作空间上变量中的元素用小括号进行索引。

对 话 框



特　　性　　　标量扩展　　　不

采样时间　　　从驱动模块中继承

状态个数　　　0

直接输出　　　是

20. Filter

图　　标

$$\frac{1}{1+2z^{-1}}$$

所在库名　　Discrete library

说　　明　　Filter 模块的功能是实现无限脉冲响应(IIR)和有限脉冲响应(FIR)滤波器。本模块接收标量输入,并产生一个标量输出。

用户可以用参数 Numerator 和 Denominator 这两个参数以 z^{-1}幂级数递增的顺序来定义分子和分母的系数,且分母的阶次必须大于或等于分子的阶次。

根据 Filter 模块对分子和分母多项式的定义的不同,在其图标上显示不同的分子和分母多项式,有关其详细的描述请参见 Transfer Fcn 模块。

参　　数　　Numerator

分子多项式系数向量,默认值为[1]。

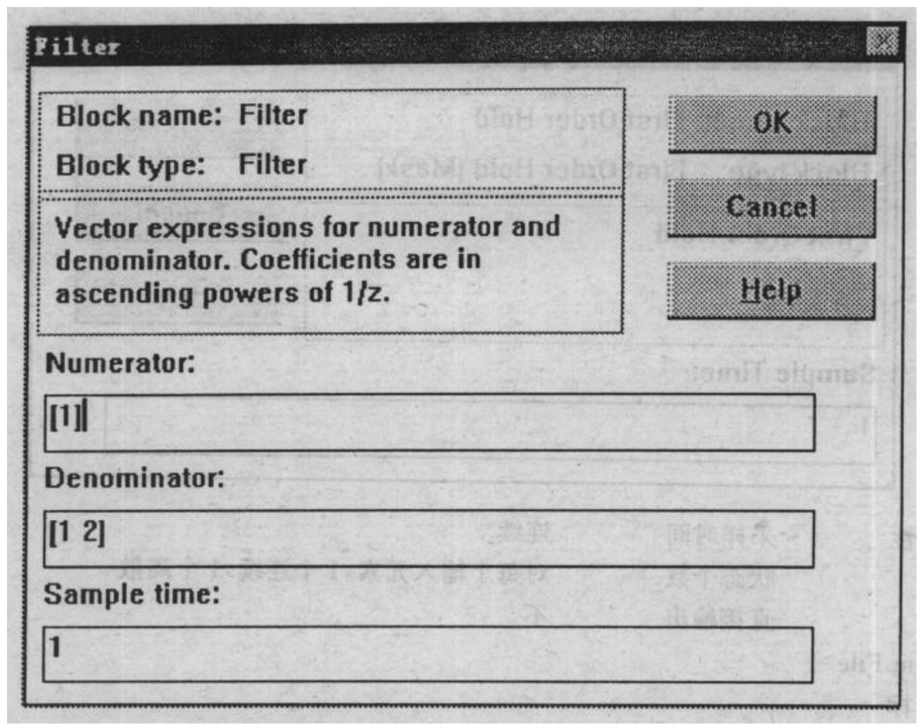
Denominator

分母多项式系数向量,默认值为[1　2]。

Sample Time

采样周期,Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期;第二个元素(如果有的话)是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟,负的偏移表采样的提前。

对 话 框



特　性	标量扩展	N/A
	采样时间	离散
	状态个数	其状态个数为分母的阶次
	直接输出	如果分子和分母阶次相同，则直接输出

21. First－Order Hold

图　标



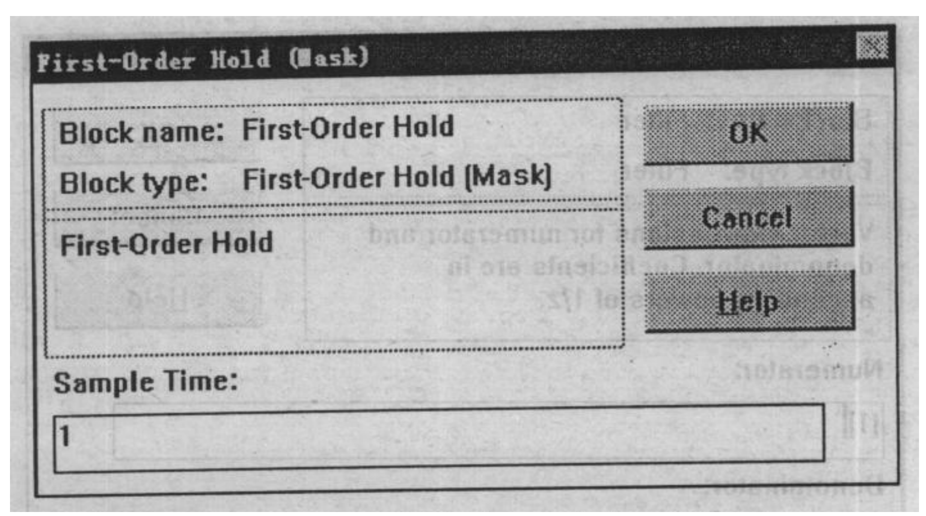
所在库名　Discrete library

说　明　First－Order Hold 模块能是在一定的时间间隔内实现一阶采样保持器。本模块接收一个输入，并产生一个输出。两者既可以是标量，也可以是向量。如果用户想比较一下零阶保持器和一阶保持器的区别，在命令窗口键入 fohdemo 即可。

参　数　Sample Time

采样周期，Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素（如果有的话）是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

对 话 框



特　　性　　采样时间　　连续

状态个数　　对每个输入元素,1 个连续,1 个离散

直接输出　　不

22. From File

图　　标

untitled.mat

所在库名　　Source library

说　　明　　From File 模块的功能是从规定的文件中读数据,并把读到的数据当作其它模块的输入。同时,在本模块的图标上显示被读文件的文件名。

这个数据文件必须包含一个至少有两行的矩阵。第一行必须是单调递增的时间,其它行的数据与第一行相应列上的时间一一对应。

本模块采用时间数据来计算其输出,但在输出中不包含时间值。这意味着一个矩阵中若包含 m 行,则本模块输出一个行数为 m－1 的向量,这个向量是由除第一行以外的数据所组成的。

如果在某一时间需要一个输出,而这个时间又处在文件中两个时间点之间,这时利用下面的线性插值公式,经插值后输出。

$$y=y_1+\frac{(t-t_1)(y_2-y_1)}{(t_2-t_1)}$$

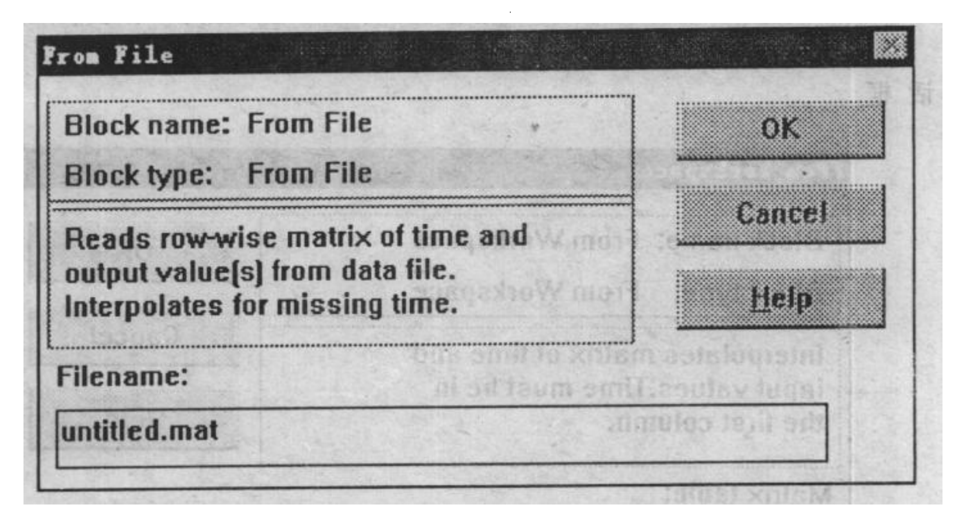
其中 t_1,t_2,y_1,y_2 分别是最接近要求时间的两个时间和两个输出。

如果要求时间小于文件中第一个时间或大于最后一个时间,SIMULINK 将分别利用最前面两点或最后面两点进行线性外插,上面计算的公式不变。

参　　数　　Filename

包含数据的文件名,默认文件名为 untitled. mat。

对话框



特　　性　　　标量扩展　　　N/A

　　　　　　　采样时间　　　从驱动模块中继承

　　　　　　　状态个数　　　0

23. From Workspace

图　　标

[T,U]

所在库名　　　Source library

说　　明　　　From File 模块的功能是从工作面里的矩阵中读数据。同时，在本模块的图标上显示被读矩阵的名称。

这个矩阵至少有两列。第一列必须是单调递增的时间，其它列上的数据与第一列相应行上时间一一对应。

本模块采用时间数据来计算其输出，但在输出中不包含时间值。这意味着一个矩阵中若包含 m 列，则本模块输出一个列数为 m－1 的向量，这个向量是由除第一列以外的数据所组成的。

如果在某一时间需要一个输出，而这个时间又处在文件中两个时间之间，这时利用下面的线性插值公式，经插值后输出。

$$y=y_1+\frac{(t-t_1)(y_2-y_1)}{(t_2-t_1)}$$

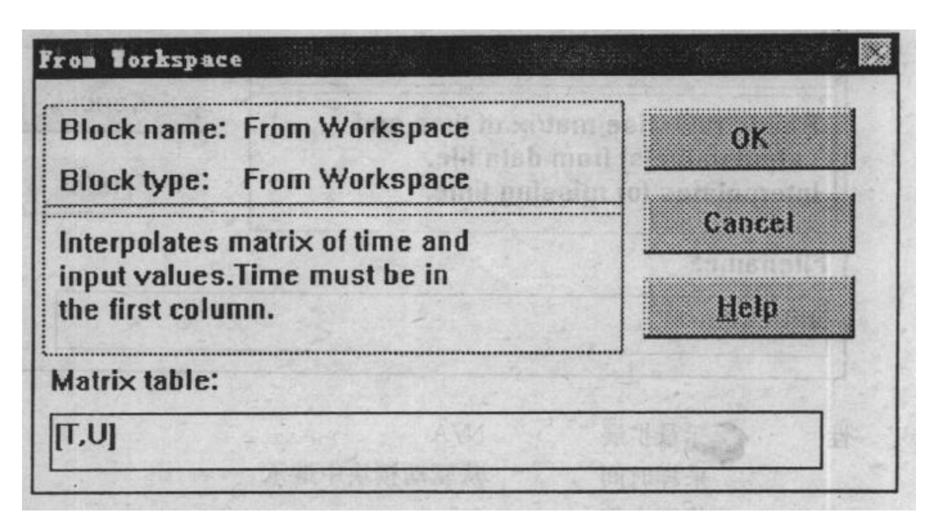
其中 t_1,t_2,y_1,y_2 是最接近要求时间的两个时间和两个输出。

如果要求时间小于文件中第一个时间或大于最后一个时间，SIMULINK 将分别利用最前面两点或最后面两点进行线性外插，上面计算的公式不变。

参　　数　　　Matrix table

包含时间和数据的矩阵。如果这些值不在同一个矩阵，必须用[T,U]来明确时间列向量 T 和数据矩阵 U。如果它们在同一个矩阵中，必须明确矩阵名，避免用 ans 来明确数据。

对话框



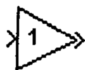
特　　性　　标量扩展　　N/A

采样时间　　从驱动模块中继承

状态个数　　0

24. Gain

图　　标



所在库名　　Linear library

说　　明　　Gain 模块的功能是对输入乘上一个规定的常数、变量或表达式后再输出。

如果模块的输入是一个标量，增益也必须是标量，输出也是个标量。

如果模块的输入是一个向量，输出必须是具有相应维数的向量。而增益既可以是标量，也可以是向量。

(1) 如果增益定义为一个标量，输入向量的每一个元素乘上这个标量，然后生成输出向量中的相应元素。

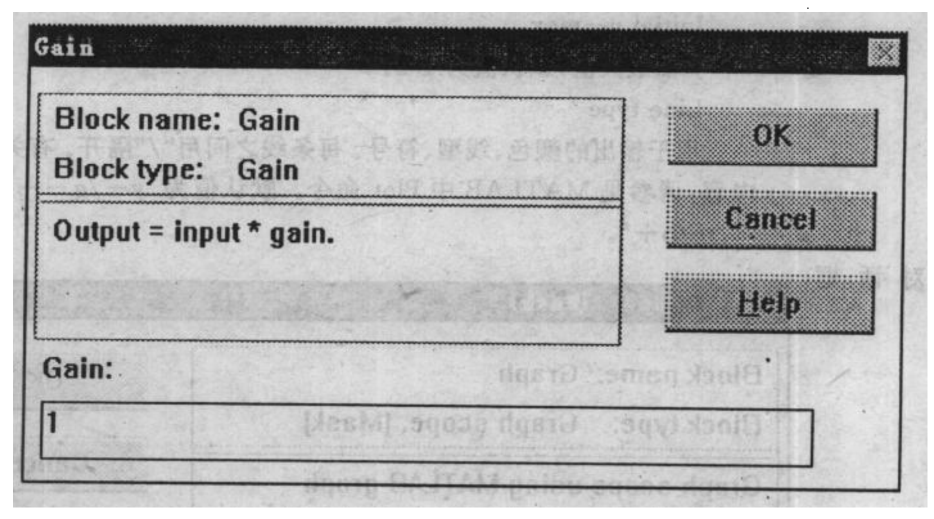
(2) 如果增益定义为一个向量，输入向量的每一个元素乘上增益向量中的相应元素，然后生成一个输出向量中的元数。在这种情况下，增益和输入向量的维数必须一致。

如果 Gain 模块足够大的话，则在 Gain 图标中显示由参数域 Gain 中输入的参数。如果输入的是一个变量，尽管这个变量在括号中定义，仍显示变量名，模块本身每次求取这个变量的值，并把这个值显示出来。如果参数 Gain 的值太长而在模块中不能显示，则在模块的图标中显示－K－。

参　　数　　Gain

增益可定义为常量、向量、变量或表达式，默认值为 1。

对 话 框



特　　性　　　标量扩展　　　参数 Gain

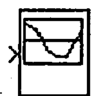
　　　　　　　采样时间　　　从驱动模块中继承

　　　　　　　状态个数　　　0

　　　　　　　直接输出　　　是

25. Graph Scope

图　　标



所在库名　　　Sinks library

说　　明　　　Graph Scope 模块的功能是在 MATLAB 图形窗口中把输入像示波器一样显示出来。

本模块接收单个输入，这个输入既可以是标量，也可以是向量，并以仿真时间为 x 轴，输入数据为 y 轴在屏幕上显示出来。最初，显示区域在 x 方向由 Initial Time Range 定义，在 y 方向由 Initial y－min 和 Initial y－max 定义。在仿真开始后，y 值根据信号的最大和最小值自动调整比例显示出来。

本模块把输入和时间数据存到一个缓冲器中，缓冲器的大小由参数 Storage points 定义，这个参数是本模块所能显示的最大数据点个数。

如果本模块的输入是一个向量，则各个量将以参数 Line type 规定的值，以不同的颜色、线型和打印符号显示出来。

参　　数　　　Time Range

在图形窗口中显示的 x 轴(时间轴)的长度。如果仿真时间超过这个范围，窗口就会被刷新，默认值为 20 s。

Initial y－min

y 轴最小值，默认值为－1.1。

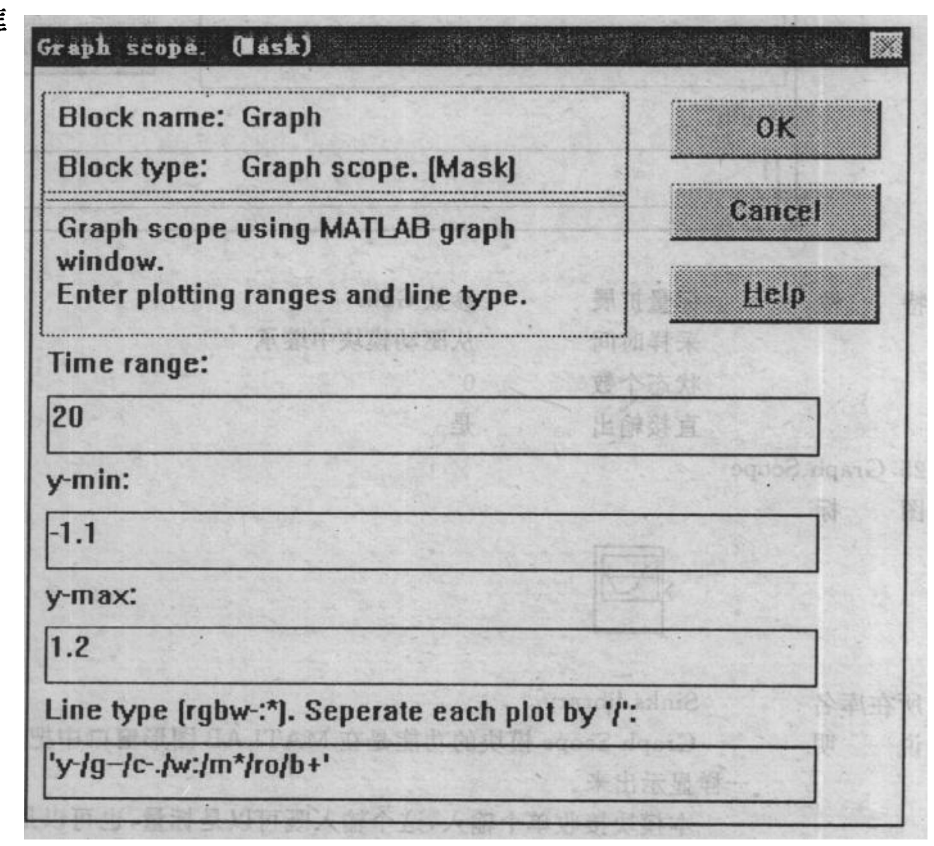
Initial y—max

y 轴最大值，默认值为 1.2。

Line type

用于输出的颜色、线型、符号。每条线之间用"/"隔开。有关这些代码的内容，请参见 MATLAB 中 Plot 命令。默认值为'y－/g－－/c－./w:/m *.ro/b＋'。

对话框

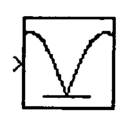


特　　性　　采样时间　　从它的驱动模块中继承

　　　　　　状态个数　　4 个离散状态(内部用来存储)

26. Hit Crossing

图　　标



所在库名　　Sinks library

说　　明　　Hit Crossing 模块的功能是当输入接近某一值时，增加仿真步长的个数，这样有利于提高某些不连续系统仿真精度。本模块采用可复位的积分

器，并允许用参数 Crossing Value 来限定容许偏差。

Hit Crossing 模块必须连接有阶跃不连续的信号。有关如何使用本模块，有一个演示程序可以参考，只要在命令窗口键入 bounce 即可。

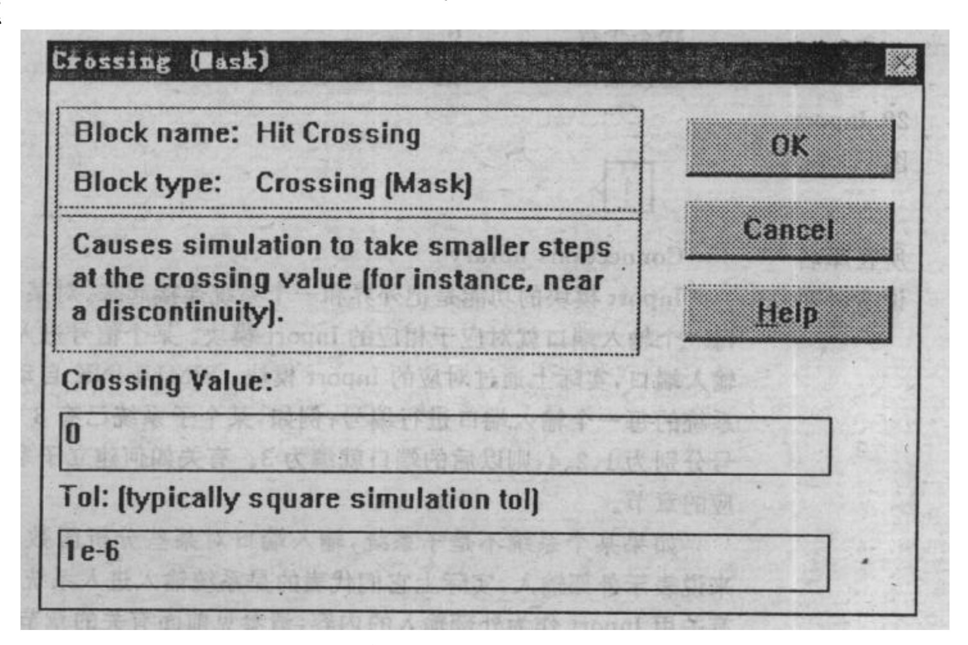
参　　数　　Crossing Value

在这一点有阶跃不连续性，默认值为 0。

Tolerance

在参数 Crossing Value 附近的容许偏差。这个值越小，在不连续点就需要积分步长越小。默认值为 1e－6。

对 话 框



特　　性		
	标量扩展	是
	采样时间	连续
	状态个数	从驱动模块中继承

27. Inner Product

图　　标



所在库名　　Linear library

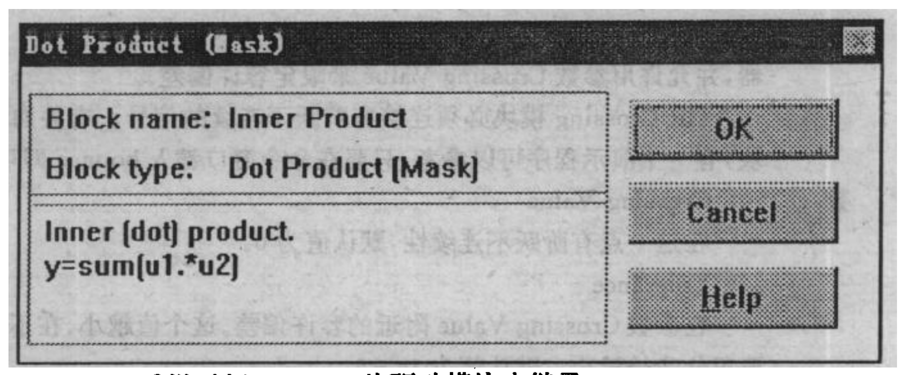
说　　明　　Inner Product 模块的功能是实现两个输入向量的点积。标量输出 y 就等价 Matlab 运算

$$y=u1' * u2$$

其中：$u1$ 和 $u2$ 分别表示向量输入。如果输入向量的维数不一致，就进行标量扩展。

要实现元素和元素的乘积而不进行求和运算，采用 Product 模块。

对 话 框



特　　性　　　采样时间　　　从驱动模块中继承
　　　　　　　状态个数　　　0
　　　　　　　直接输出　　　是

28. Inport

图　　标

所在库名　　　Connections library

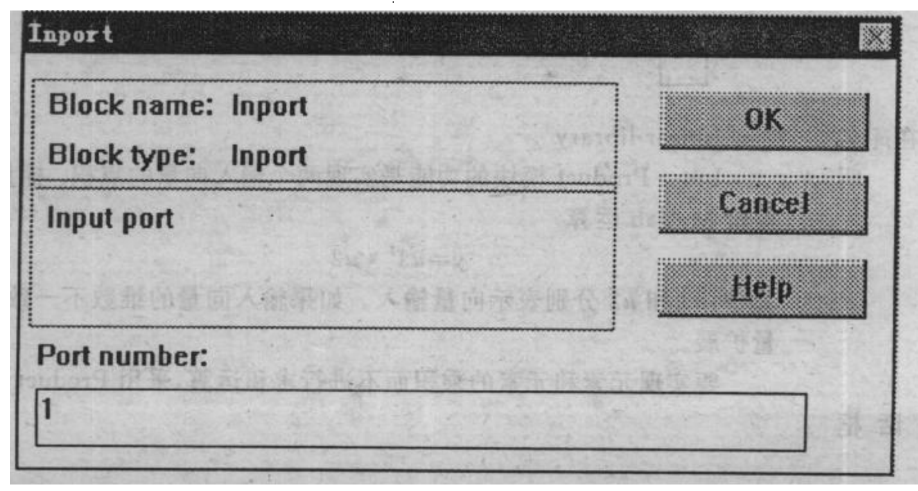
说　　明　　　Inport 模块的功能是把外界和一个系统连接起来。对某个子系统来说，每一个输入端口就对应于相应的 Inport 模块。某个信号进入某个子系统的输入端口，实际上通过对应的 Inport 模块。SIMULINK 自动从 1 开始给子系统的每一个输入端口进行编号；例如，某个子系统已有 3 个输入端口，编号分别为 1、2、4，则以后的端口就编为 3。有关如何建立子系统请看前面相应的章节。

如果某个系统不是子系统，输入端口对某些分析函数像 linmod、trim 来说表示外部输入，实际上它们代表的是系统输入进入系统的那些输入点。有关用 Inport 作为外部输入的内容，请参见前面有关的章节。

参　　数　　　Port number

输入端口的个数。尽管输入端口的编号可以改变，SIMULINK 仍然自动给每个口进行编号。

对 话 框



特　　性　　　采样时间　　　从驱动模块中继承

　　　　　　　状态个数　　　0

　　　　　　　直接输出　　　在子系统中直接输出

29. Integrator

图　　标　　　1/s

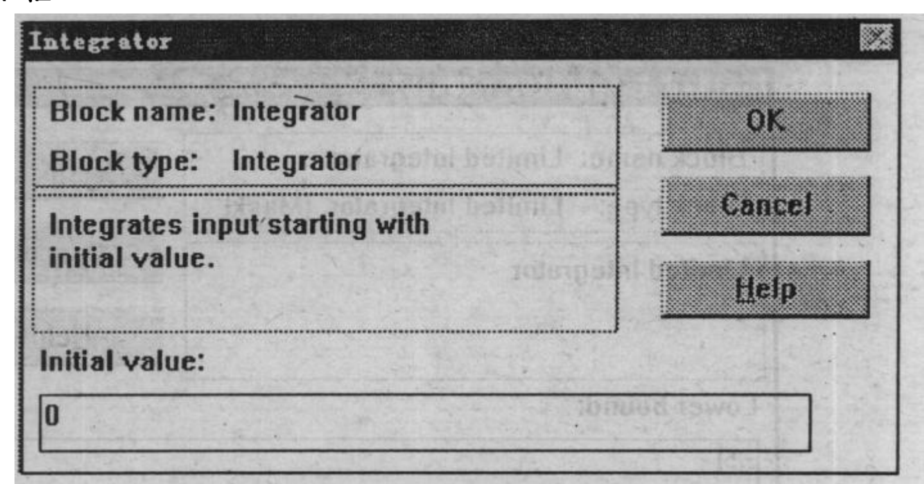
所在库名　　　Linear Library

说　　明　　　Integrator 模块的功能是对输入进行积分。本模块接收的输入信号，既可以是标量，也可以是向量。而输出信号的维数与输入信号一致。

参　　数　　　Initial value

初始条件既可以是一个常量，也可以是一个标量。如果输入是一个向量，初始条件也是个向量，那么它们的维数应一致。如果初始条件是一个标量，则进行标量扩展。输入向量中第 m 个元素就以初始值向量中的第 m 个元素为初始条件进行积分，生成输出向量中的第 m 个元素。

对 话 框



特　　性　　　标量扩展　　　对参数 Initial value 进行扩展

　　　　　　　采样时间　　　连续

　　　　　　　状态个数　　　输入的个数

　　　　　　　直接输出　　　不

30. Limited Integrator

图　　标　　　1/s

所在库名　　　Nonlinear library

说　　明　　　Limited Integrator 模块的功能是对输入信号进行积分，并限制积分输出超过规定的值。本模块的输出由下列条件来决定：

(1) 当积分值小于下界参数,则输出的是下界。

(2) 当积分值介于上界与下界参数之间,则输出的是积分值。

(3) 当积分值大于上界参数,则输出的是上界。

本模块的设计目的是当积分值不在规定的范围之内,取消积分作用,从而有效地防止积分饱和。

本模块接收一个输入,并产生一个输出。两者既可以是标量,也可以是向量。

参　　数　　Lower bound

积分值的下界。

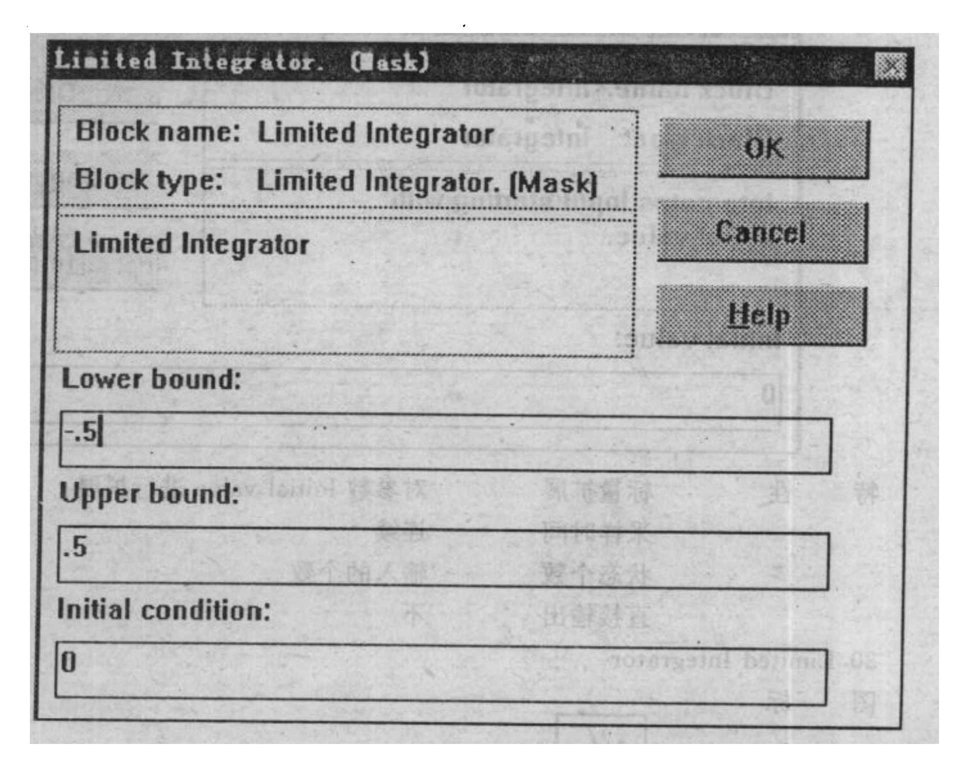
Upper bound

积分值的上界。

Initial Condition

初始条件既可以是一个常量,也可以是一个标量。如果输入是一个向量,初始条件也是个向量,那么它们的维数应一致。如果初始条件是一个标量,则进行标量扩展。输入向量中第 m 个元素就以初始值向量中的第 m 个元素为初始条件进行积分,生成输出向量中的第 m 个元素。

对 话 框



特　　性

标量扩展	参数
采样时间	离散
状态个数	从驱动模块或参数中继承
直接输出	否

31. Logical Operator

图　　标

AND

所在库名　　Nonlinear library

说　　明　　Logical Operator 模块的功能是对输入信号进行 AND、NAND、OR、NOR、NOT、XOR 逻辑运算。每一个输入既可以是向量，也可以是标量。

本模块的输出不仅取决于输入的个数、向量的维数，而且取决于逻辑算子。对于设定的逻辑算子会在图标上显示出来。

如果输入信号有两个或两个以上时，本模块就在所有的输入信号之间完成运算。如果输入信号是个向量，运算就在向量中对应的元素之间进行，并产生一个向量输出。

如果是单个向量输入，除了逻辑算子 NOT 以外，本模块就对向量中的所有元素进行逻辑运算。

逻辑算子 NOT 只接受单个输入，这个输入既可以是标量，也可以是向量。

如果输入是一个向量，则本模块对向量中的每一个元素进行逻辑补运算，然后输出。

参　　数　　Operator

对输入信号进行逻辑运算的算子。有效的算子如前面所列出的这些逻辑算子。

Number of Input Ports

模块输入的个数。

对 话 框

Logical Operator

Block name: Logical Operator

Block type: Logical Operator

Output = Input1 OP Input2 OP ...
Operators include AND, OR,
NAND, NOR, XOR, and NOT.

OK

Cancel

Help

Operator:

AND

Number of Input Ports:

2

特　　性　　　标量扩展　　　输入
　　　　　　　采样时间　　　从驱动模块或参数中继承
　　　　　　　状态个数　　　0
　　　　　　　直接输出　　　是

32. Look－Up Table

图　　标

所在库名　　Nonlinear library

说　　明　　Look－Up Table 模块的功能是对输入进行分段线性映射成一张由模块参数所定义成的一张表。

用户可以通过参数 Vector of input values 和 Vector of output values 来定义这张表。本模块通过比较输入信号和模块输入向量中的值，然后产生一个输出值。

如果输入信号和模块输入向量中的值相等，则输出的是输出向量中对应的元素。

如果输入信号与模块输入向量中的任何值都不相等，则在表对应的两个元素之间进行线性插值，然后输出。如果模块的输入小于第一个或大于最后一个输入向量中的元素，本模块就分别在最前面两个或最后面两个元素之间进行外插。

如果输入向量中有两个相同的值，而对应的输出值不同，这可以通过对这个相同的输入向量中的元素定义两次，每一次只对应一个输出即可。

例如，假如你定义的输入和输出的参数为：

Vector of input values：[0 1 1 2]

Vector of output values：[－1 1 1 1]

本模块就产生如图 9－1 所示的输入和输出关系。

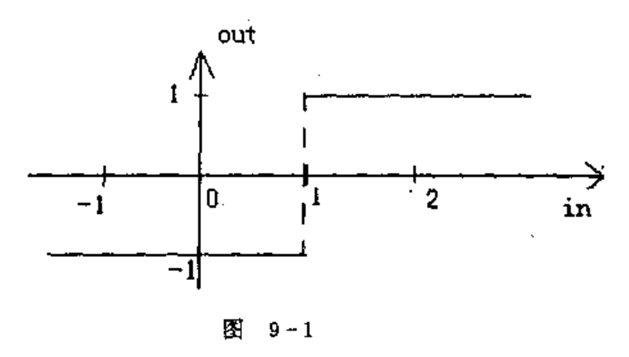


图　9－1

当输入小于 1 时，输出为－1；当输入大于 1 时，输出为 1。

Look－Up Table 模块在其图标上显示输入向量和输出向量之间的关系图。当用户改变输入和输出的参数时，一旦你用鼠标单击OK 按钮来关闭对话框后，输入和输出之间的关系图就会重画。

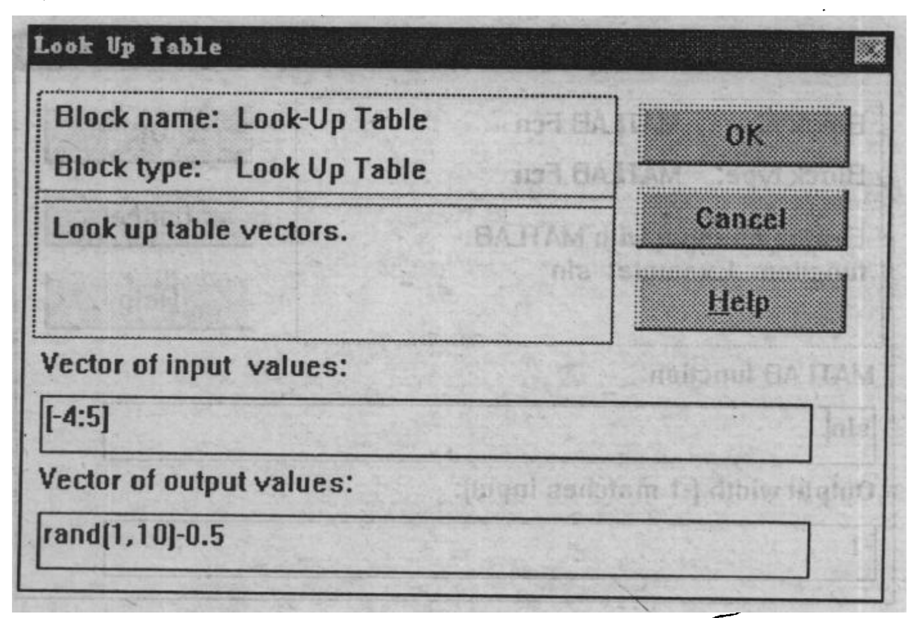
参　　数　　Vector of input values

这个向量包含模块可能要输入的值，这个向量和输出向量的维数必须一致，而且必须是单调递增。

Vector of output values

这个向量包含模块的输出值，这个向量和输入向量的维数必须一致。

对 话 框



特　　性　　标量扩展　　参数

采样时间　　从驱动模块或参数中继承

状态个数　　0

直接输出　　是

33. MATLAB Fcn

图　　标

MATLAB Function

所在库名　　Nonlinear library

说　　明　　MATLAB Fcn 模块的功能是对输入信号进行规定的 MATLAB 函数或表达式处理。本模块接受一个输入，它既可以是向量，也可以是标量。本模块对输入信号进行规定的函数或表达式运算后，然后按规定的输出维数进行输出。对本模块来说，下面这些表达式是有效的：

```
sin
atans(u(1),u(2))
u(1)^ u(2)
```

参　　数　　MATLAB Fcn

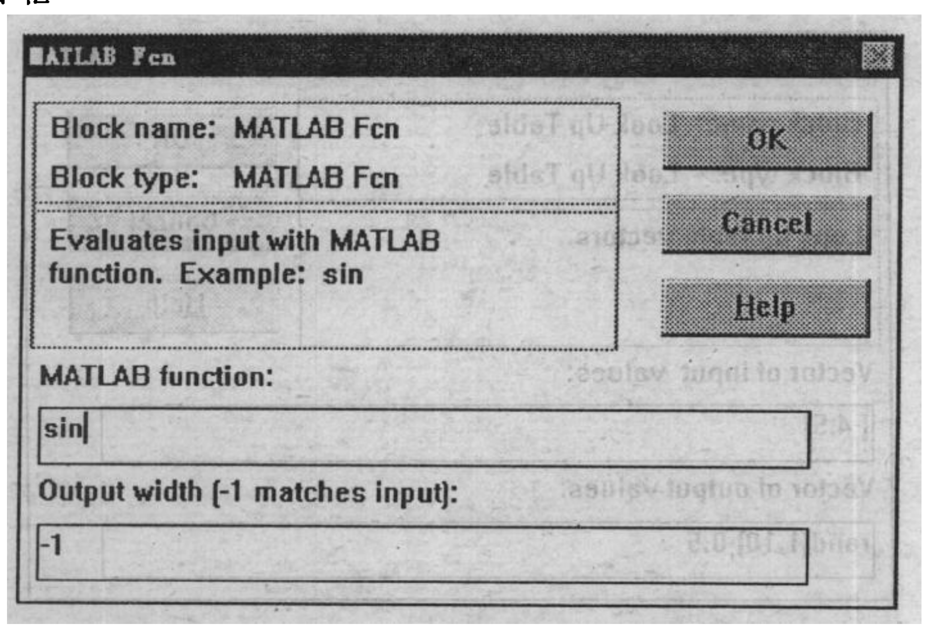
函数或表达式。如果只定义了一个函数，那么在函数中就没有必要包含

输入变量。例如，如果规定的函数为 sin，那么输出的是 sin(u)，其中 u 是输入。

Output width

输出维数。如果输出的维数等于输入的维数，规定为－1；否则的话，用户必须明确规定其正确的维数。

对 话 框



特　　性	采样时间	从驱动模块或参数中继承
	状态个数	0
	直接输出	是

34. Matrix Gain

图　　标

所在库名　　Linear library

说　　明　　Matrix Gain 模块的功能是实现一个矩阵增益。它把向量输入乘上一个规定的增益矩阵后，然后产生一个输出：

$$y = Ku$$

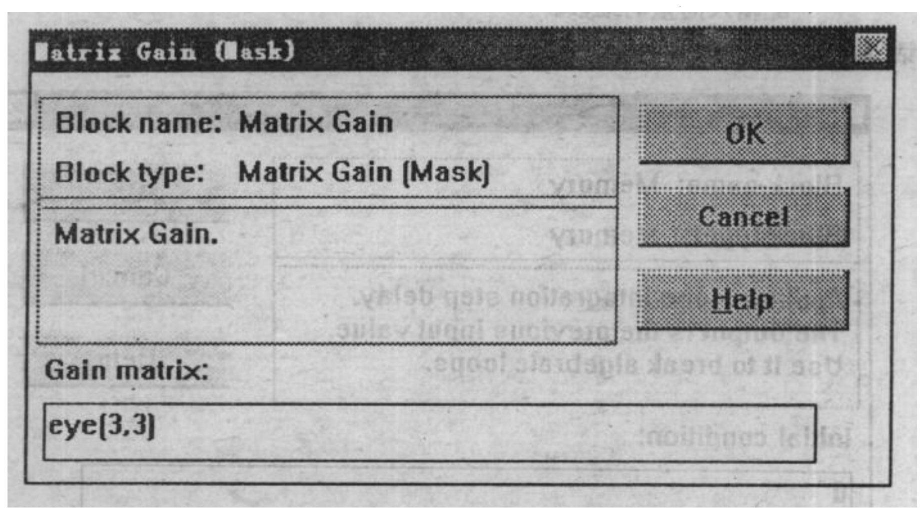
其中 K 为增益矩阵，u 为输入向量。

如果规定增益矩阵有 m 行 n 列，则输入向量应该是有 n 个元素的向量，输出向量就是有 m 个元素的向量。

参　　数　　Gain matrix

增益矩阵。

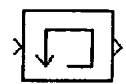
对 话 框



特　　性	标量扩展	不
	采样时间	从驱动模块或参数中继承
	状态个数	0
	直接输出	是

35. Memory

图　　标



所在库名　　Nonlinear library

说　　明　　Memory 模块的功能是对输入信号进行一步的采样保持，然后输出前一步的输入值。本模块接受一个输入，并产生一个输出，它既可以是标量，也可以是向量。本模块可用来取消代数环。例如，下面这个例子用来演示在仿真中如何产生阶跃信号，如图 9－2 所示。

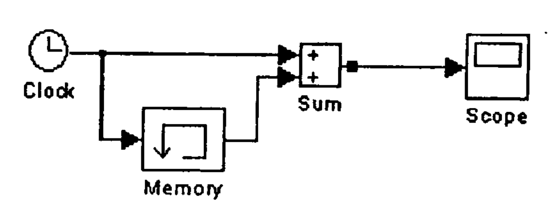


图　9－2

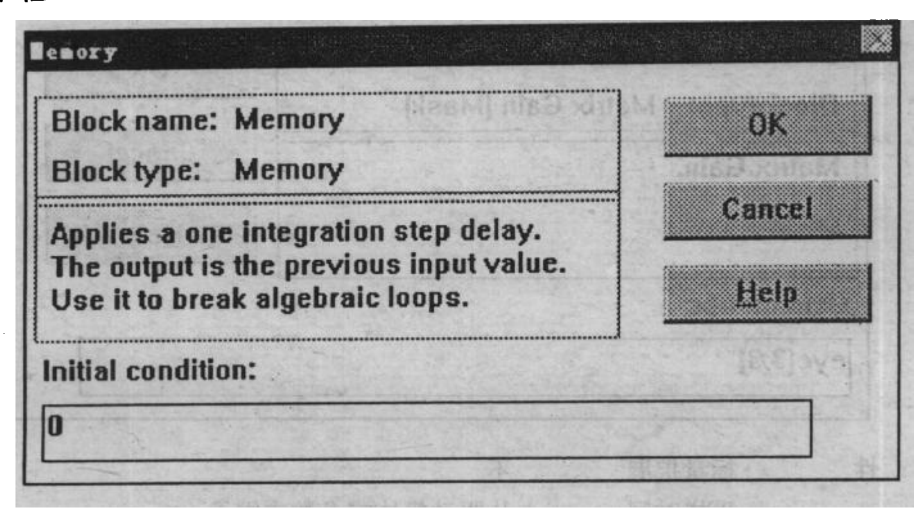
Sum 模块把从 Clock 模块中产生的当前时间减去 Memory 保存的上一步时间，然后输出给 Scope 模块进行显示。

参　　数　　Initial condition

在初始积分时的输出。如果必要的话，SIMULINK 进行标量扩展来匹

配输入向量的维数。

对 话 框



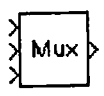
特　　性　　采样时间　　连续

状态个数　　0

直接输出　　不

36. Mux

图　　标



所在库名　　Connections library

说　　明　　Mux 模块的功能是把几个输入合并成一个向量输出。本模块接受规定数目的输入，它既可以是标量，也可以是向量。Mux 模块的输出为一个向量。

SIMULINK 会根据其输入的个数自动调整其输入端口的数目，并在图标上显示出来。

参　　数　　Number of inputs

输入的个数和维数。输入信号维数的总数必须和输出的维数一致。

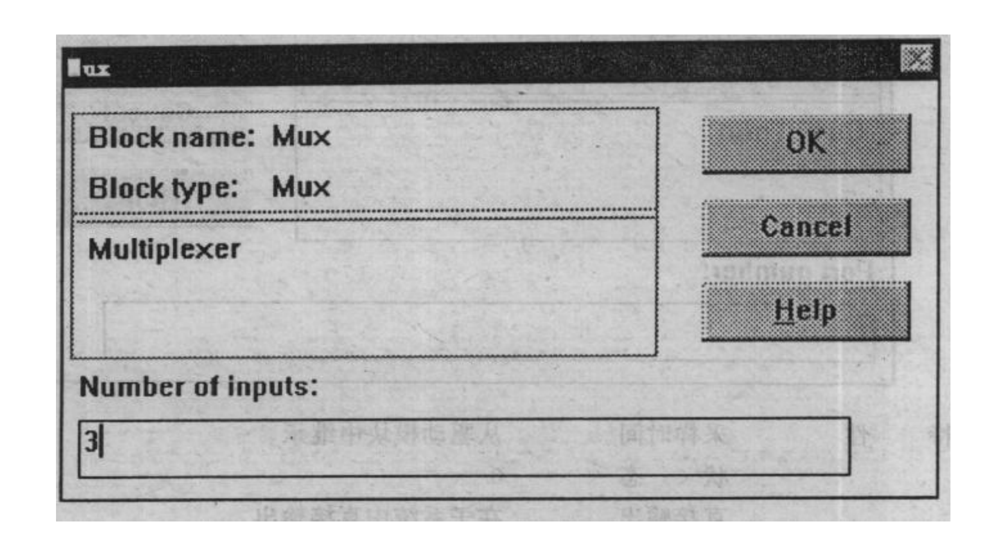
一般来说，规定输入端口的个数，SIMULINK 会根据各个输出端口输出给 Mux 模块的情况来决定其维数。如果确有必要，要明确规定各个输入的维数，用户可以把它们的维数定义为一个向量。对于仿真时需要动态决定其维数的输入，其对应的元素为－1。

例如，[4 1 2]表示三个输入组成一个具有 7 个元素的输出向量：前面的 4 个输出来自于第一个输入，第五个输出来自于第二个输入，第六和第七个输出来自于第三个输入。如果对这些输入有没有固定的元素不太重要的话，用户可以规定参数 Number of Input 为 3。

如果规定有三个输入且第一个输入向量的元素个数为 4 个，用户可以

把参数设为[4 −1 −1]。SIMULINK 会根据第二和第三个输入向量的维数，相应地决定输出的维数。

对 话 框



特 性	采样时间	从它的驱动模块中继续
	状态个数	0
	直接输出	不

37. Output

图 标

[1]

所在库名 Connections library

说 明 Output 模块的功能建立一个系统与外界进行联系的桥梁。

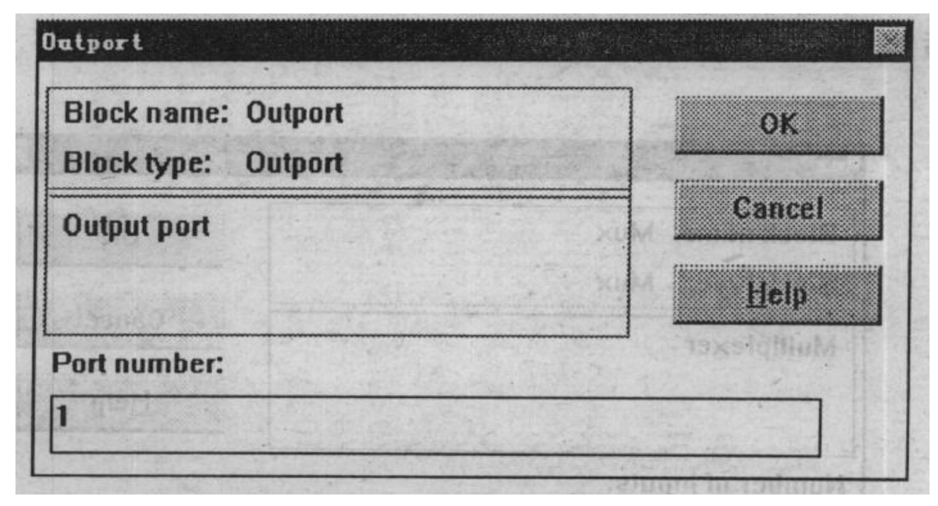
对一个子系统来说，在子系统模块上的每一个输出端口对应于一个子系统的 Output 模块。SIMULINK 会自动给子系统的每一个 Output 模块从 1 开始进行编号。对新增加的 Output 模块，它的编号为第一个可用的号码。例如，假如已有 Output 模块的编号为 1、2、4，那么下一个 Output 模块的编号就为 3。有关子系统的建立方法，请参见第六章。

如果某个系统不是子系统，Output 模块对某些分析函数像 linmod、trim 和积分器来说定义了系统输出。有关 Output 模块这种的使用方法，请参见第七章。

参 数 Port number

输出端口的个数。尽管输出端口的编号可以改变，SIMULINK 仍然自动给每个端口进行编号。

对 话 框



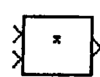
特　　性　　采样时间　　从驱动模块中继承

状　　态　　0

直接输出　　在子系统中直接输出

38. Product

图　　标



所在库名　　Nonlinear library

说　　明　　Product 模块的功能是对每一个输入进行乘积运算，然后输出。其输入既可以是标量，也可以是向量。用户可以用 Product 模块实现标量相乘，向量相乘，一个或多个向量与相同的标量相乘及单个向量元素之间的相乘：

(1) 如果是标量输入，输出是输入的乘积。

(2) 如果是向量输入，输出是由输入向量各对应元素的乘积所组成的向量，这时所有输入向量的维数必须一致。

(3) 如果是标量和向量输入，SIMULINK 首先自动把标量扩展为和其它向量具有相同维数的向量，然后相乘。输出是由输入向量各对应元素的乘积所组成的向量。

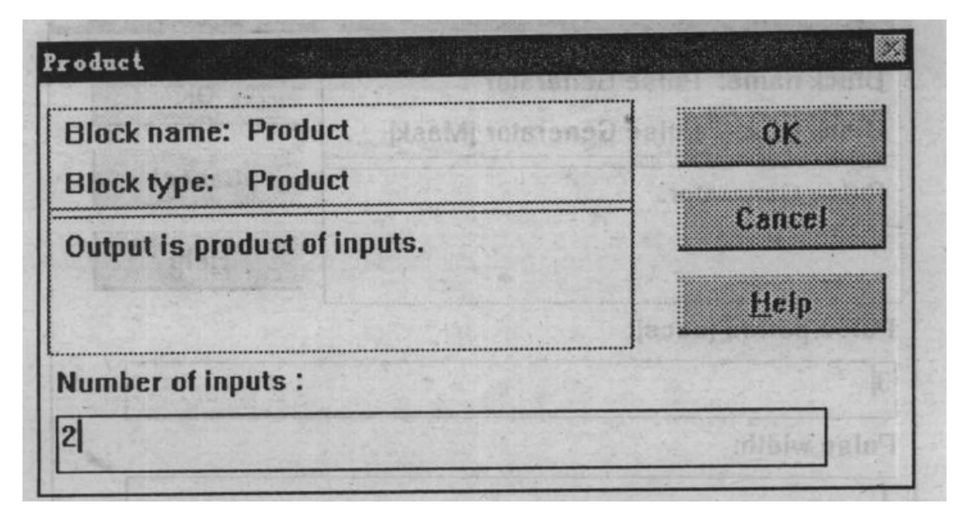
(4) 如果输入的是单个向量，Product 模块把输入向量中的元素进行乘积运算，在这种情况下在图标上显示连乘符号。

如果有必要的话，SIMULINK 会重新调整模块的大小来显示所有的输入端口。

参　　数　　Number of inputs

输入到模块的个数。

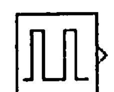
对 话 框



特　　性	采样时间	从驱动模块中继承
	状　　态	0
	直接输出	在子系统中直接输出

39. Pulse Generator

图　　标



所在库名　　Source library

说　　明　　Pulse Generator 模块的功能是在规定的时间间隔内产生一系列脉冲信号。本块模产生一个标量输出。

参　　数　　Pulse period

所产生脉冲的周期，单位为 s。默认值为 3 s。

Pulse width

所产生脉冲的宽度。默认值为 2 s。

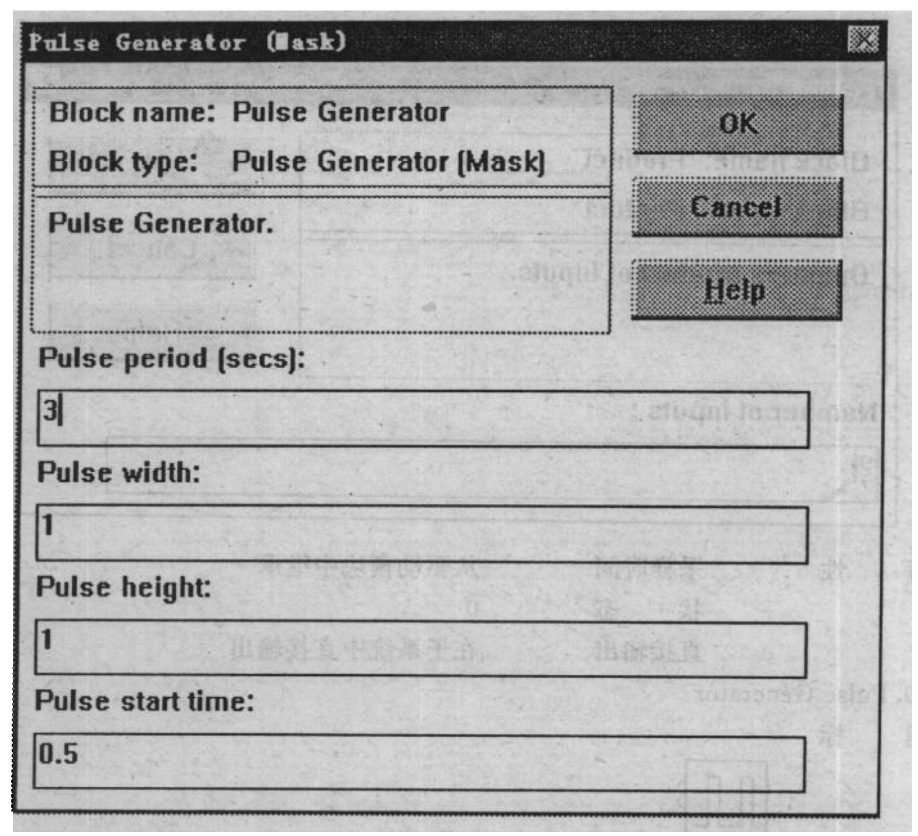
Pulse height

所产生脉冲的高度。如果规定为一个向量，本模块产生一个脉冲向量。默认值为 1。

Pulse start time

脉冲产生时的时延，单位为 s。默认值为 0.5 s。

对 话 框

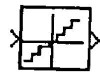


特　　性　　　　采样时间　　　连续

　　　　　　　　离散状态　　　是

40. Quantizer

图　　标



所在库名　　　Nonlinear library

说　　明　　　Quantizer 模块的功能是对输入信号通过一个台阶型的函数后，把输入轴上的相临点映射成输出轴上一个点。它把一个平滑输入信号变成一个台阶型的输出信号。输出按照四舍五入或最接近舍入法来产生一个对原点对称的输出：

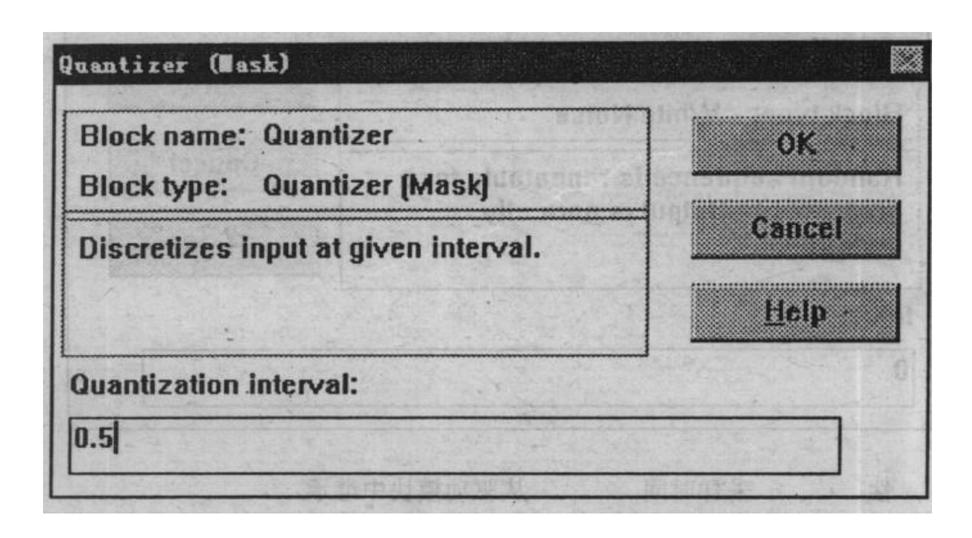
$$y=q*round(u/q)$$

其中 y 是输出，u 是输入，q 是由参数 Quantization Interval 确定。其输入和输出既可以是标量，也可以是向量。

参　　数　　　Quantization Interval

量化间隔，也称为量化因子。Quantizer 模块的允许输出值为 n*q，其中 n 是一个整数，q 是 Quantization Interval。

对话框



特　　性	标量扩展	参数
	采样时间	从驱动模块或参数中继承
	状　　态	0
	直接输出	是

41. Random Number

图　　标



所在库名　　Sources library

说　　明　　Random Number 模块的功能是产生正态分布的随机数。对于任何给定的种子，它和 MATLAB 函数 randn 一样产生相同的随机数序列。每次仿真开始时，种子被重新设置为所需要的值。

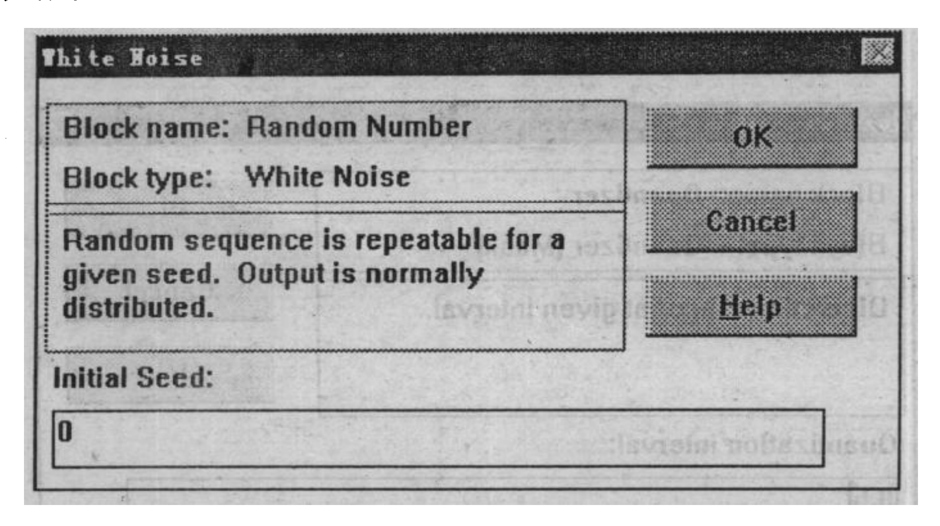
Random Number 模块是一个伪随机、正态分布的随机数发生器。这个随机数序列具有均值为零、方差为 1。这个系列是可重复的，而且可以用相同的种子由 Random Number 模块来产生。要产生一个随机数向量，可定义参数 Initial seed 为一个向量。

由于计算上的问题，本模块不宜与连续系统连用，这时可用 Band Limited White Noise 模块来替代。

参　　数　　Initial Seed

用来产生随机数的种子。如果定义为一个向量，本模块产生一个随机噪声信号向量。默认值为 0。

对 话 框


White Noise

Block name: Random Number

Block type: White Noise

Random sequence is repeatable for a given seed. Output is normally distributed.

Initial Seed:

0

OK

Cancel

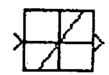
Help

特　　性　　　采样时间　　　从驱动模块中继承

　　　　　　　状　　态　　　0

42. Rate Limiter

图　　标



所在库名　　　Nonlinear library

说　　明　　　Rate Limiter 模块的功能是限制输入信号的一阶导数，使输出信号的变化速率不能快于规定的限制值。其一阶导数是用下面的式子来计算：

$$rate=\frac{\Delta u}{\Delta t}=\frac{u(i)-y(i-1)}{t(i)-t(i-1)}$$

其中 $u(i)$和 $t(i)$分别是当前的输入和时间。而 $y(i-1)$和 $t(i-1)$分别是前一步的输出和时间。本模块的输出要根据 rate 和模块中参数 Rising slew rate、Falling slew rate 之间的关系来确定。

(1) 如果 *rate* 大于参数 Rising slew rate 的绝对值，输出可由下式来计算：

$$y(i)=\Delta t(abs(R))+y(i-1)$$

其中 $y(i)$是模块的当前输出，R 是参数 Rising slew rate。

(2) 如果 *rate* 小于参数 Falling slew rate，输出可由下式来计算：

$$y(i)=\Delta t(abs(F))+y(i-1)$$

其中 F 是参数 Falling slew rate。

(3) 如果 *rate* 介于 R 和 F 之间，输出可由下式来计算：

$$y(i)=\Delta t * \text{rate}+y(i-1)$$

本模块接受一个输入，并产生一个输出，两者既可以是标量，也可以是

向量。

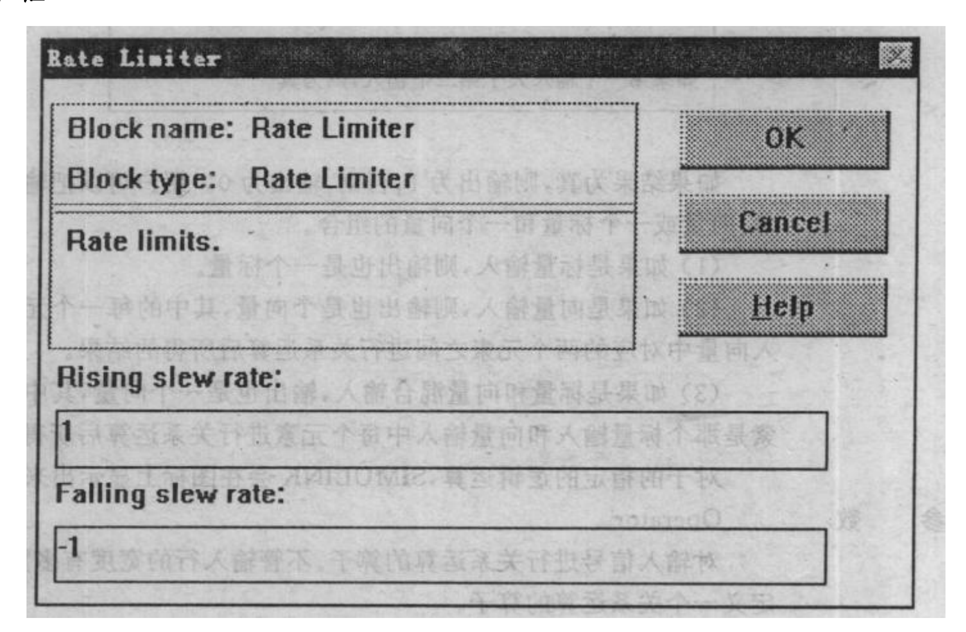
参　　数　　Rising Slew rate

对幅值不断增大的输入信号的导数进行限制。这个参数既可以是标量，也可以是向量。

Falling Slew rate

对幅值不断减小的输入信号的导数限制。这个参数既可以是标量，也可以是向量。

对 话 框



特　　性　　标量扩展　　参数

采样时间　　从驱动模块中继承

状　　态　　0

直接输出　　是

43. Relational Operator

图　　标

所在库名　　Nonlinear library

说　　明　　Relational Operator 模块的功能是对两个输入信号完成关系运算，并根据下面这张表产生相应的输出。

算子	输出
==	如果两个输入相等，则为真
!=	如果两个输入不等，则为真
<	如果第一个输入小于第二个输入，则为真
<=	如果第一个输入小于或等于第二个输入，则为真
>=	如果第一个输入大于或等于第二个输入，则为真
>	如果第一个输入大于第二个输入，则为真

如果结果为真，则输出为1；否则，输出为0。用户可以把输入定义为标量、向量或一个标量和一个向量的组合。

(1) 如果是标量输入，则输出也是一个标量。

(2) 如果是向量输入，则输出也是个向量，其中的每一个元素是两个输入向量中对应的两个元素之间进行关系运算后所得的结果。

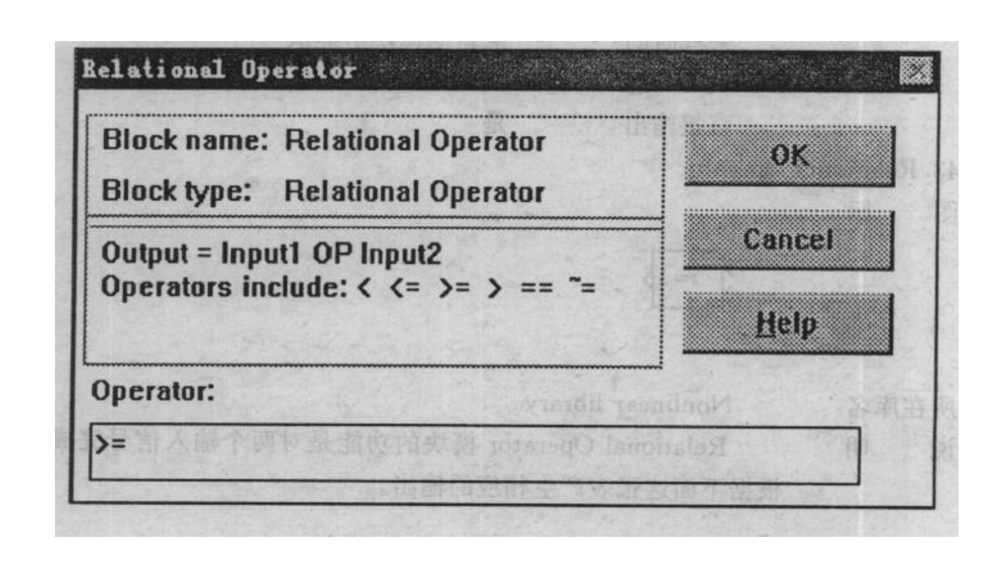
(3) 如果是标量和向量混合输入，输出也是一个向量，其中的每一个元素是那个标量输入和向量输入中每个元素进行关系运算后所得的结果。

对于的指定的逻辑运算，SIMULINK 会在图标上显示出来。

参　数　　Operator

对输入信号进行关系运算的算子。不管输入行的宽度有多宽，用户只能定义一个关系运算的算子。

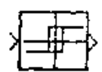
对话框



特　　性　　　标量扩展　　　输入
　　　　　　　采样时间　　　从驱动模块中继承
　　　　　　　状　　态　　　0
　　　　　　　直接输出　　　是

44. Relay

图　　标



所在库名　Nonlinear library

说　　明　Relay 模块的功能是在规定的两个值之间进行切换。当 Relay 打开时，它一直保持打开状态，直到输入信号下降到小于参数 Input for off 所规定的值。当 Relay 关闭时，它一直保持关闭，直到输入信号上升到超过参数 Input for on 所规定的值。

本模块接受一个输入，并产生一个输出，两者既可以是标量，也可以是向量。

参　　数　Input for on

打开继电器所需的阀值。当输入大于这个值时，继电器就一直打开。如果定义参数 Input for on 大于参数 Input for off，则 Relay 模块模拟一个磁滞特性；如果把这两个参数定义为相同值，则 Relay 模块模拟一个具有该阀值的开关。

Input for off

关闭继电器所需的阀值。当输入小于这个值时，继电器就一直关闭。如果定义参数 Input for off 大于参数 Input for on，则 Relay 模块就在两个输出之间、以系统仿真的速率来回进行切换。

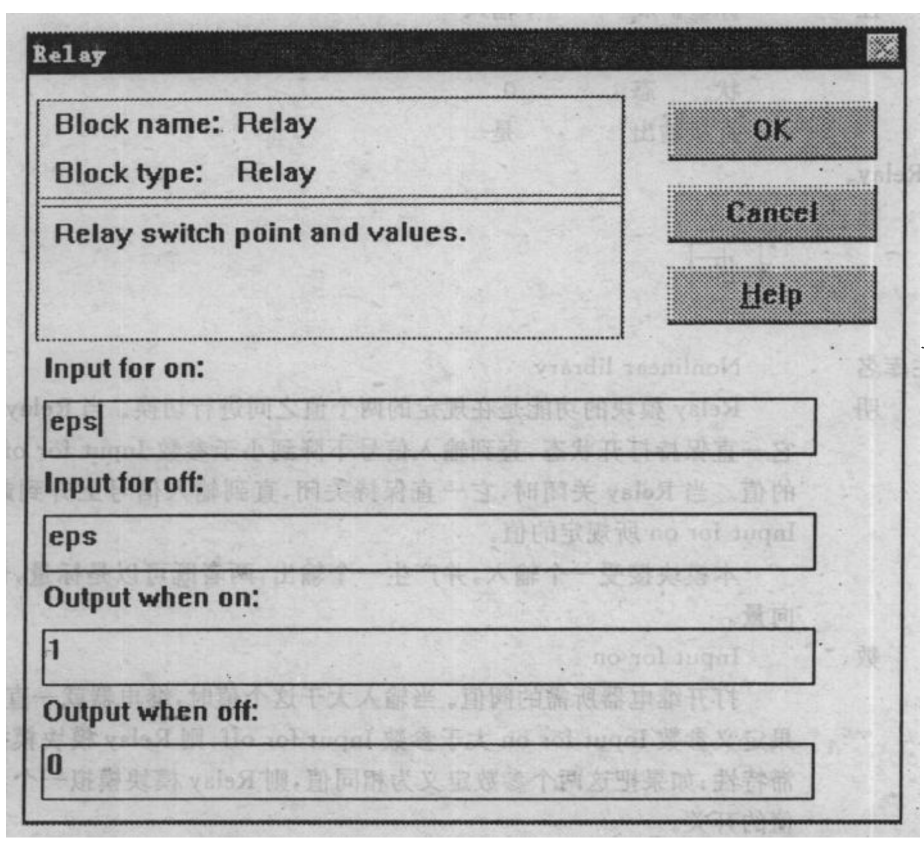
Output when on

当继电器打开时的输出值。

Output when off

当继电器关闭时的输出值。

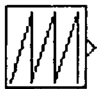
对 话 框



特　　性	标量扩展	参数
	采样时间	从驱动模块中继承
	状　　态	0
	直接输出	是

45. Repeating Sequence

图　　标



所在库名　　Source library

说　　明　　Repeating Sequence 模块的功能是有规律地产生一个任意的周期信号。Repeating Sequence 模块是采用一维 Look - Up Table 模块、通过在点与点之间进行线性插值后来实现。

本模块产生一个标量输出。

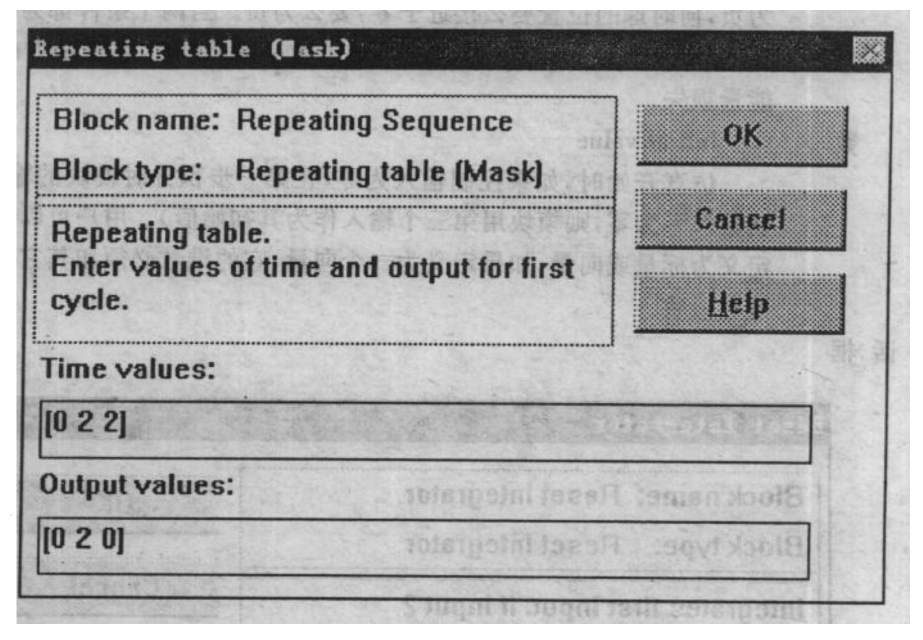
参　　数　　Time values

单调递增的时间向量。默认值为[0 2 2]。

Output values

输出向量。每一个输出向量的元素与相应的时间向量中的元素相对应。默认值为[0 2 0]。

对 话 框



特　　性　　　采样时间　　　连续

　　　　　　　状　　态　　　0

46. Reset Integrator

图　　标

所在库名　　　Nonlinear library

说　　明　　　Reset Integrator 块是一个在仿真期间，其状态可重新设置的 Integrator 模块。本模块有三个输入：

(1) 信号

(2) 控制或需重新设置时的逻辑表达式，当这个逻辑表达式为真(或不为零)，就使其状态重新设置。

(3) 重新设置时其状态的初始值。

当控制输入为零时，本模块就对输入信号进行积分。当控制输入不为零时，本模块就把其状态设为与第三个输入对应的那个值，然后重新对输入信号进行积分。本模块只在主要的积分步长内对控制输入进行检测，而在其它时间内，本模块就和 Integrator 模块一样对输入信号进行积分。

本模块的输入和单个输出，既可以是向量，也可以是向量。

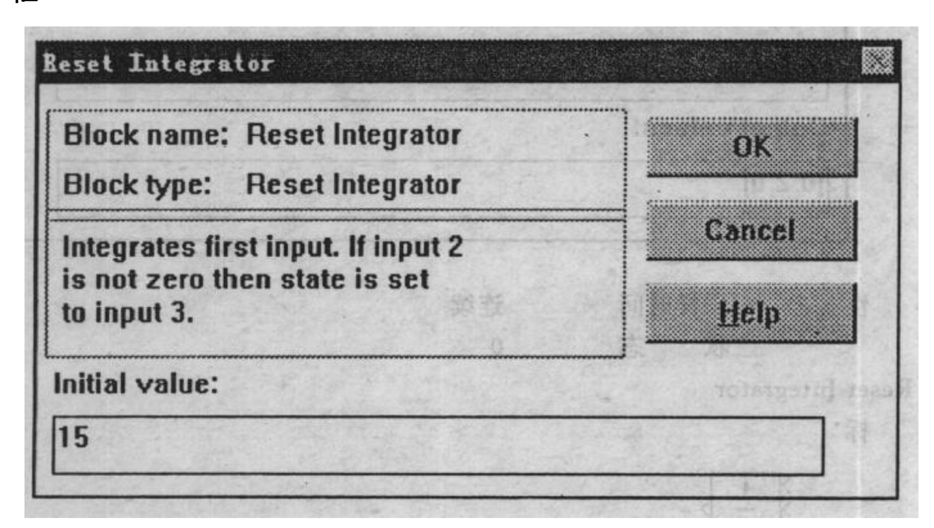
有一个有关弹性球模型的演示程序采用了本模块，要运行这个演示程序，只需在 MATLAB 的命令窗口中键入 bounce 即可。在这个模型中，Reset Integrator 模块对第一个输入(重力)进行积分后来产生球的速度，而

Integrator 模块用来产生球的位置。一个 Fcn 模块用来决定何时球的速度为负，何时球的位置要么接近于零，要么为负。当两个条件都为真时，Reset Integrator 模块的状态就重新设为 $-0.8*$ 前面的值，它表示方向的变化和能量损失。

参　数　　Initial value

仿真开始时，如果控制输入为零，在第一步积分时模块的输出（如果控制输入不为零，则模块用第三个输入作为其初始值）。用户可以把这个参数定义为标量或向量；如果定义为一个向量，它的维数必须和其它输入向量的维数一致。

对 话 框

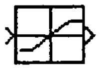


特　性

标量扩展	输入和参数 Initial value
采样时间	连续
状　　态	从驱动模块或参数 Initial value 中继承
直接输出	否

47. Saturation

图　标



所在库名　　Nonlinear librry

说　明　　Saturation 模块的功能是对输入信号强行施加上界和下界。当输入信号在参数 Lower output limit 和 Upper output limit 定义的范围内时，输入信号不作任何变化直接输出；而在这个范围之外，输入信号就限制为其上界或下界。

当上面这两个参数定义为相同值时，则模块就输出这个值；如果参数 Upper output limit 设为小于 Lower output limit，则模块输出下界值。

本模块接收一个输入信号，并产生一个输出信号，它们既可以是标量，

也可以是向量。

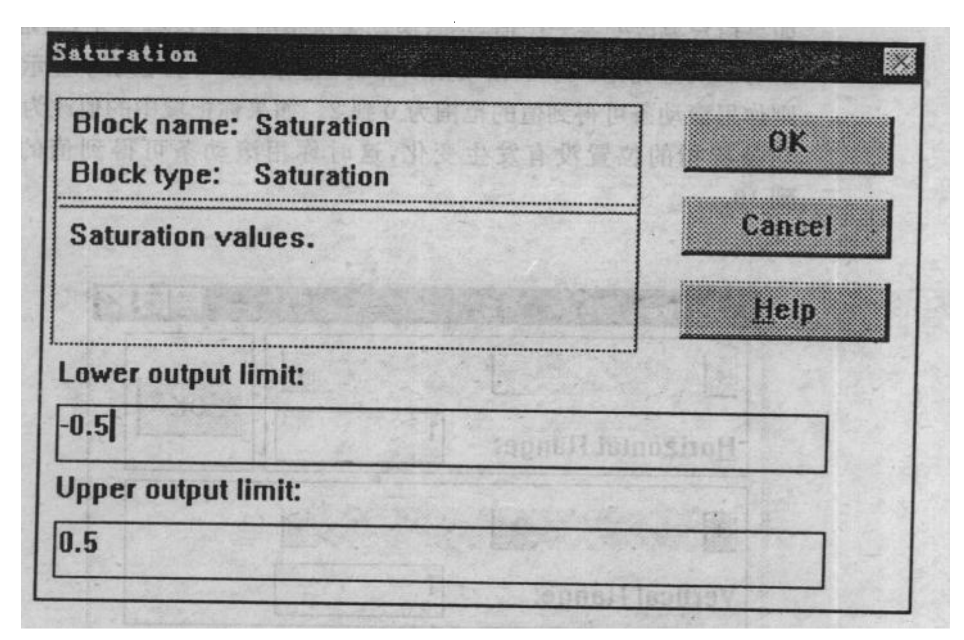
参　　数　　Lower output limit

作用于输入信号的下界。如果输入信号小于这个值，则输出这个下界。

Upper output limit

作用于输入信号的上界。如果输入信号大于这个值，则输出这个上界。

对 话 框



特　　性　　标量扩展　　参数

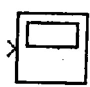
采样时间　　从驱动模块中继承

状　　态　　0

直接输出　　是

48. Scope

图　　标



所在库名　　Sinks library

说　　明　　Scope 模块的功能是显示输入信号和仿真时间之间的关系曲线。本模块接收一个输入，它既可以是标量，也可以是向量。Scope 模块允许用户调整时间轴和输入值的显示范围，也可以移动和更改 Scope 窗口的大小。不过，减小其窗口，可能导致水平参数域和垂直参数域及 OK 按钮消失。

当用户做一个仿真时，SIMULINK 不打开 Scope 模块的窗口。在仿真期间，用户可以修改模块中的参数值。

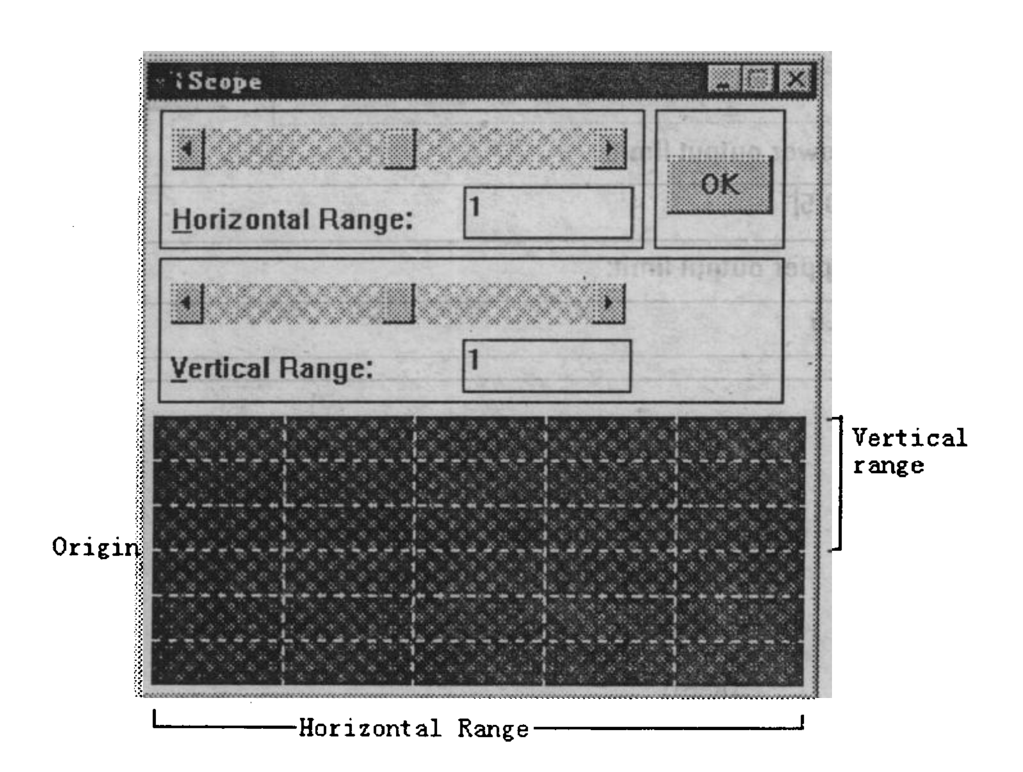
参　　数　　Vertical range

在显示区域正负方向可以用来显示的范围。正如下图所示的那样，整个

显示范围是参数 Vertical range 所设值的两倍。

Horizontal range

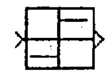
在显示区被更新以前，所能显示的最大秒数。Scope 模块的窗口如下图所示，图中标出了参数 Vertical range 和 Horizontal range 的意义，用户可以看到在相应区域内可以设置这两个参数，也可以通过滚动条来设置这两个参数。滚动条所指的位置总是和其域中显示的值匹配，且其下界总为零。如果用户要改变某一个值，尽管滚动条所指的位置没有变化，但是其值的范围可以发生变化。例如，滚动条所指的位置是在中间，在域中显示的值为 1，则你用滚动条可得到值的范围为 0 到 2。如果你把域中的值改为 5，尽管滚动条所指的位置没有发生变化，这时你用滚动条可得到值的范围为 0 到 10。



特　性	采样时间	从驱动模块中继承
	状　态	0

49. Sign

图　标



所在库名　　Nonlinear library

说　明　　Sign 模块的功能是输出输入信号的符号，其方法如下：

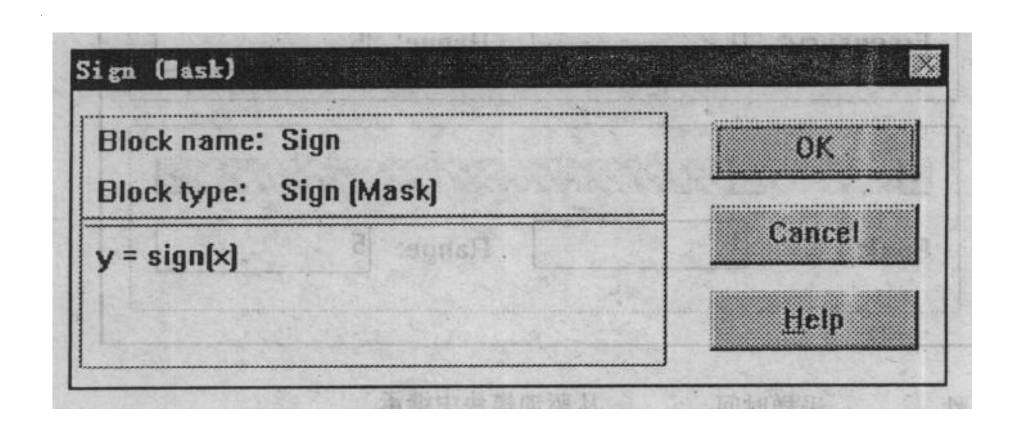
(1) 如果输入大于零，则输出为 1。

(2) 如果输入等于零，则输出为 0。

(3) 如果输入小于零，则输出为 −1。

输入和输出既可以是标量，也可以是向量。

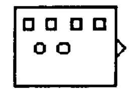
对话框



特　性	采样时间	从驱动模块中继承
	状　态	0
	直接输出	是

50. Signal Generator

图　标



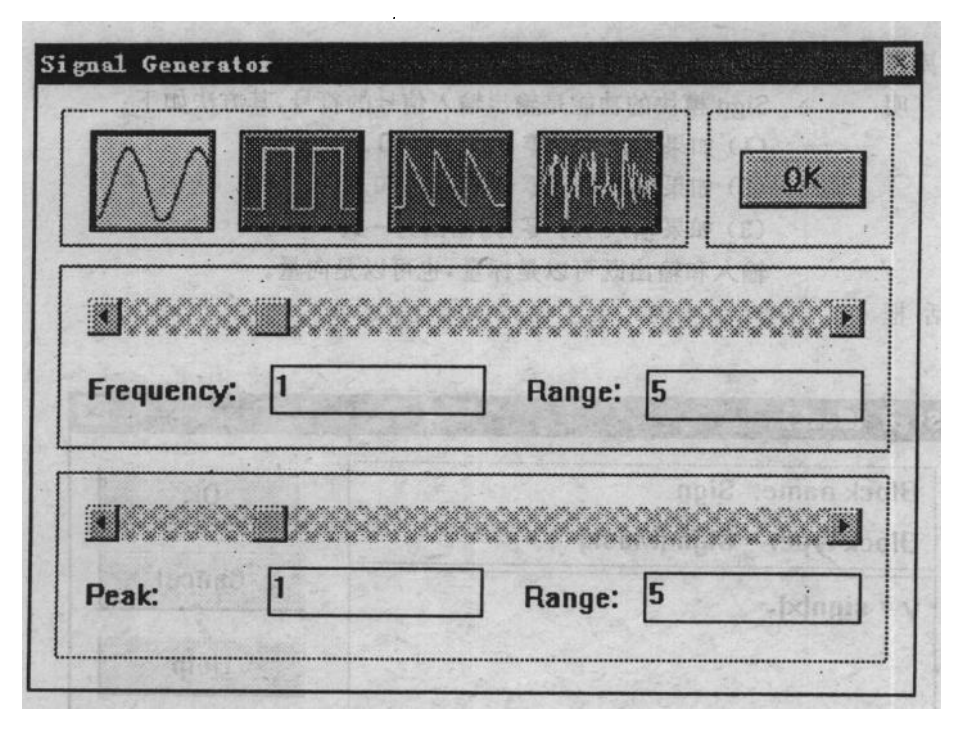
所在库名　　Sources library

说　明　　Signal Generator 模块的功能是产生 4 种不同波形的信号：正弦波、方波、锯齿波、随机信号。

参　数　　Frequency

可用滚动条或在域中输入的办法来设定信号的频率。频率的单位为 rad/s。用户只能在域中输入数字，而不能是变量。其默认值为 1rad/s 。

对话框



特　　性　　　采样时间　　　从驱动模块中继承

　　　　　　　状　　态　　　0

51. Sine Wave

图　　标



所在库名　　　Source library

说　　明　　　Sine Wave 模块的功能是产生一个时变的正弦波。本模块可以在连续和离散两种模式下工作。

Sine Wave 模块的输出由下面的公式来决定。

$$y = Amplitude \times \sin(frequency \cdot time + phase)$$

Sine Wave 模块可产生一个标量或向量输出。

在离散模式下使用 Sine Wave 模块：

如果用户设置参数 Sample Time 为一个不为零的值，本模块就像驱动一个 Zero—Order Hold 模块一样，其采样时间就为这个值。

在这种方式下使用本模块，用户可以用正弦波作为输入来为纯粹的离散系统、而不是混合系统建模。由于混合模型比离散模型更复杂，因此需要更长的时间来进行仿真。

在连续模式下使用 Sine Wave 模块：

在连续模式下使用本模块，随着时间 t 的增大，其精度损失也越大，因此就使得 Sine Wave 模块变得不精确。而在离散模式下使用，由于 Sine Wave 模块采用增量算法而不是基于绝对时间的办法，从而有效地解决其精度损失问题。因此这对需要运行很长的模型，像振荡、疲劳测试来说，就非

常有用。

增量算法，它是根据正弦波上一个采样周期的值来计算当前值的算法，这种算法采用下面的等式：

$$\sin(t+\Delta t)-\sin(t)\cos(\Delta t)+\sin(\Delta t)\cos(t)$$

$$\cos(t|\Delta t)=\cos(t)\cos(\Delta t)+\sin(t)\sin(\Delta t)$$

这些等式写成矩阵形式为

$$\begin{bmatrix}\sin(t+\Delta t)\\ \cos(t+\Delta t)\end{bmatrix}=\begin{bmatrix}\cos(\Delta t) & \sin(\Delta t)\\ -\sin(\Delta t) & \cos(\Delta t)\end{bmatrix}\begin{bmatrix}\sin(t)\\ \cos(t)\end{bmatrix}$$

既然 Δt 是常数，因此下面的矩阵也是个常数。

$$\begin{bmatrix}\cos(\Delta t) & \sin(\Delta t)\\ -\sin(\Delta t) & \cos(\Delta t)\end{bmatrix}$$

因此求 $\sin(t+\Delta t)$ 的问题就变成了把 $\sin(t)$ 乘上一个常数矩阵问题。

参　数　Amplitude　信号的幅值。默认值为 1。

Frequency　信号的频率，单位为 rad/s。默认值为 1 rad/s。

Phase　相位偏移，单位是弧度。默认值为 0 rad。

Sample Time

当用户把参数 Sample Time 设为一个不为零的值，本模块就变成离散的模块。默认值为 0。

对 话 框

Sine Wave

Block name: Sine Wave
Block type: Sine Wave

ZOH(Amp*sin(Freq*Time+Phase))

OK
Cancel
Help

Amplitude:
1
Frequency (rads/sec):
1
Phase (rads):
0
Sample Time: (Zero or empty for continuous)

特　　性　　标量扩展　　参数

采样时间　　连续或离散

状　　态　　0

52. Slider Gain

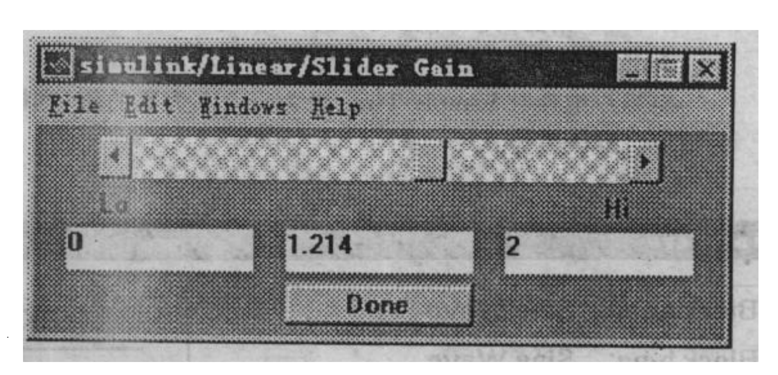
图　　标

1.317

所在库名　　Linear library

说　　明　　Slider Gain 模块的功能是在仿真期间允许用户改变标量增益。本模块接收一个输入，并产生一个输出，两者既可以是标量，也可以是向量。如果输入信号是个向量，则增益要进行标量扩展，并作用于输入向量中的每一个元素。

对 话 框



对话框包括一个滚动条、一个文本区和一个 Done 按钮。文本区从左到右分别指出下界、当前值和上界。用户可以有两种方式来改变增益：一种是通过改变滚动条位置，另一种在当前值域内键入新值，然后用鼠标单击 Done 按钮关闭对话框即可。

如果用户用鼠标在滚动条的箭头上单击一下，当前值将大约改变滚动条范围的 1%。

如果用户在滚动条长方形区域内单击一下，当前值将大约改变滚动条范围的 10%。

特　　性　　标量扩展　　参数

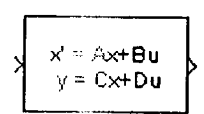
采样时间　　从驱动模块中继承

状　　态　　0

直接输出　　是

53. State - Space

图　　标



所在库名　　Linear library

说　　明　　State－Space 模块的功能是实现一个具有下列状态空间模型的系统：

$$\dot{x}=Ax+Bu$$
$$y=Cx+Du$$

其中 x 是状态向量，u 是输入向量，y 是输出向量。

本模块接收一个输入，并产生一个输出，两者既可以是标量，也可以是向量。

参　　数　　A,B,C,D

状态空间模型中的系数矩阵。

(1)A 必须是一个 n * n 的矩阵，其中 n 是状态的个数。

(2)B 必须是一个 n * m 的矩阵，其中 m 是输入的个数。

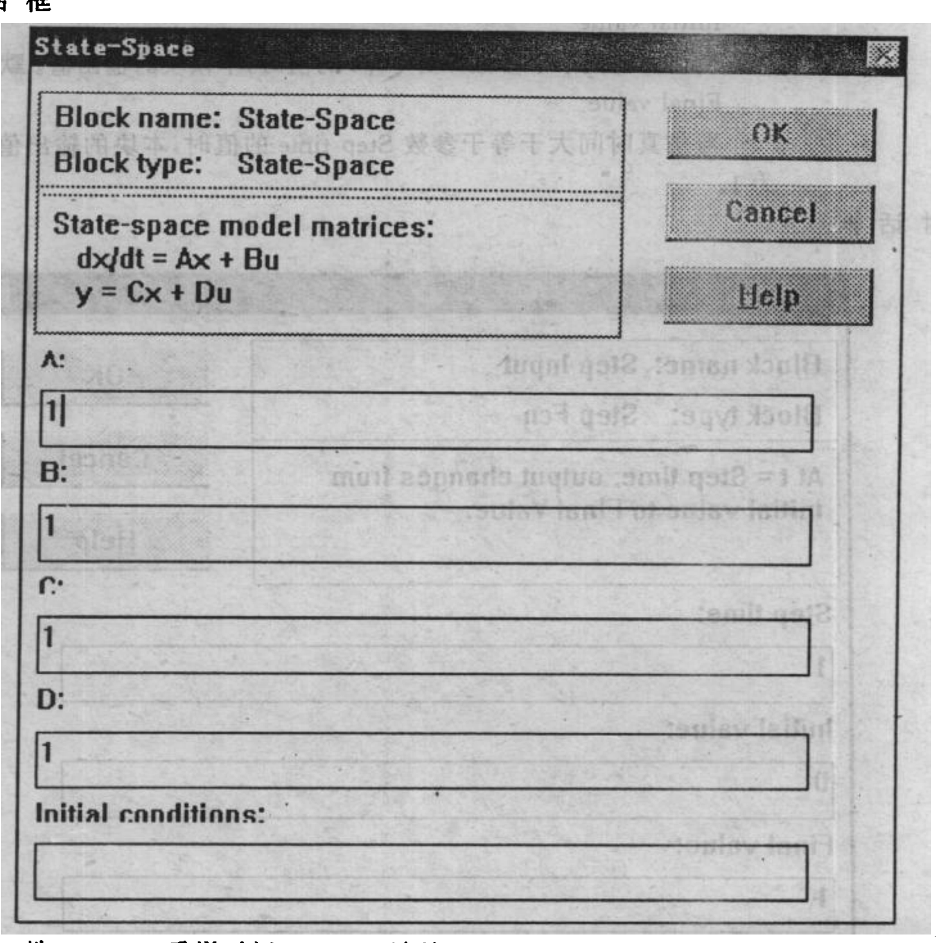
(3)C 必须是一个 r * n 的矩阵，其中 r 是输出的个数。

(4)D 必须是一个 r * m 的矩阵。

Initial conditions

初始状态向量。如果没有输入，则认为 0。

对 话 框



特　　性　　采样时间　　　连续

状　　态　　　矩阵 A 的维数

直接输出　　　只要 D! =0

54. Step Input

图　　标



所在库名　　Source library

说　　明　　Step Input 模块的功能是在某一规定时刻在两值之间产生一个跳变。如果仿真时间小于参数 Step time 的值，本模块输出的是参数 Initial value 的值。如果仿真时间大于或等于参数 Step time 的值，输出的是参数 Final value 的值。

本模块产生一个标量或向量输出，这取决于参数的维数。

参　　数　　Step time

输出参数 Initial value 的值跳变到参数 Final value 的值的时间，单位为 s。默认值为 1 s。

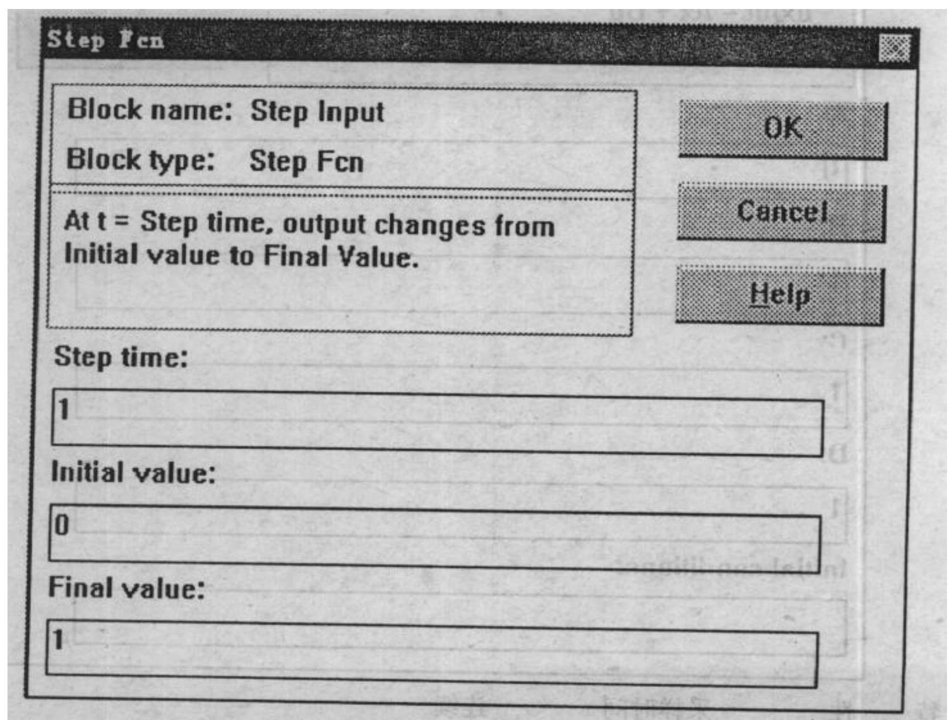
Initial value

当仿真时间小于参数 Step time 的值时，本模块的输出值。默认值为 0。

Final value

当仿真时间大于等于参数 Step time 的值时，本块的输出值。默认值为 1。

对 话 框



特　　性　　　标量扩展　　　参数

　　　　　　　采样时间　　　从驱动模块中继承

　　　　　　　状　　态　　　0

55. Stop Simulation

图　　标

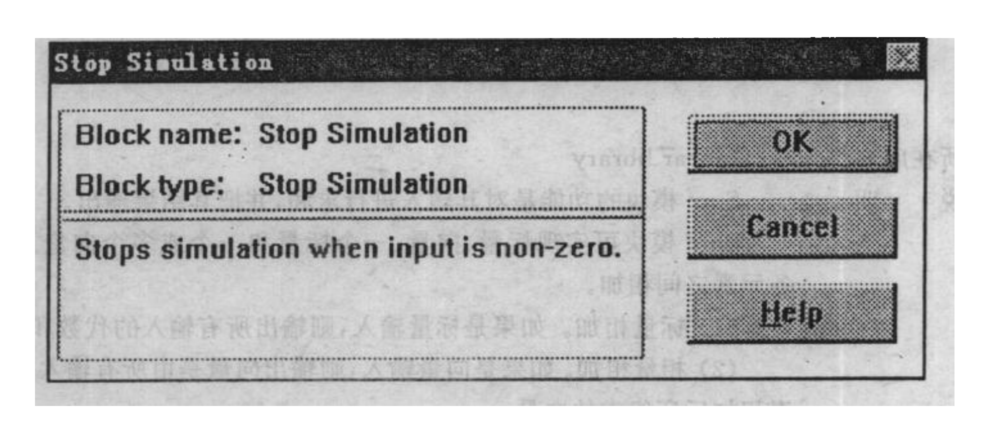
STOP

所在库名　　　Sinks library

说　　明　　　Stop Simulation 模块的功能是当输入不为零时，停止仿真。一旦模块的输入不为零，Stop Simulation 模块能迅速中断仿真。本模块可接收单个标量或向量输入。

用户可以把本模块和 Relational Operator 模块结合起来使用来控制何时终止仿真。当用户把本块和 Derivative 模块结合起来使用，那么就有可能在上升和下降的转折点上终止仿真。

对 话 框

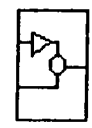


特　　性　　　采样时间　　　从驱动模块中继承

　　　　　　　状　　态　　　0

56. Subsystem

图　　标



所在库名　　　Connections

说　　明　　　Subsystem 模块用来表示一个在其它系统中的系统。用定义方框的办法可把一组模块和连线转化为一个 Subsystem 模块，其方法是首先选中要转化为 Subsystem 模块的对象，然后在 Options 菜单中选择 Group 命令。有

关建立子系统的详细信息请参见第三章。

当用户建立一个子系统时,子系统与外边这些模块之间的连线在子系统中将分别用输入和输出端口来代替。在 Subsystem 模块的图标上显示的输入端口的个数与子系统中包含的输入口的个数一致。最上面的输入连到第一个输入口,接下来那个输入连到第二个输入口,以此类推。

同样,在 Subsystem 模块图标上显示的输出端口的个数与子系统中包含的输出口的个数一致。

如果子系统中输入端口和输出端口的个数发生改变,则在图标上会重新显示输入端口和输出端口正确的个数。如果有必要的话,SIMULINK 会自动调整模块图标的大小。

对话框　　无

特　　性　　标量扩展　　可变

　　　　　　采样时间　　可变

　　　　　　状　　态　　可变

　　　　　　直接输出　　可变

57. Sum

图　　标

所在库名　　Linear library

说　　明　　Sum 模块的功能是对其输入进行求和,并把其结果输出。

Sum 模块可实现标量、向量、一个标量和一个或多个向量及单个向量各元素之间相加。

(1) 标量相加。如果是标量输入,则输出所有输入的代数和。

(2) 相量相加。如果是向量输入,则输出向量是由所有输入向量对应元素相加后所组成的向量。

(3) 一个标量与向量相加。如果是标量和向量输入,则本模块进行标量扩展。每一个标量就自动扩展为具有相应维数的向量,且其中的各个元素都等于这个标量。

(4) 一个向量中的各元素相加。如果是单个向量输入,则本模块就对输入向量中的所有元素进行相加,然后产生一个标量输出。在这种情况下,SIMULINK 自动把图标改为求和算子。

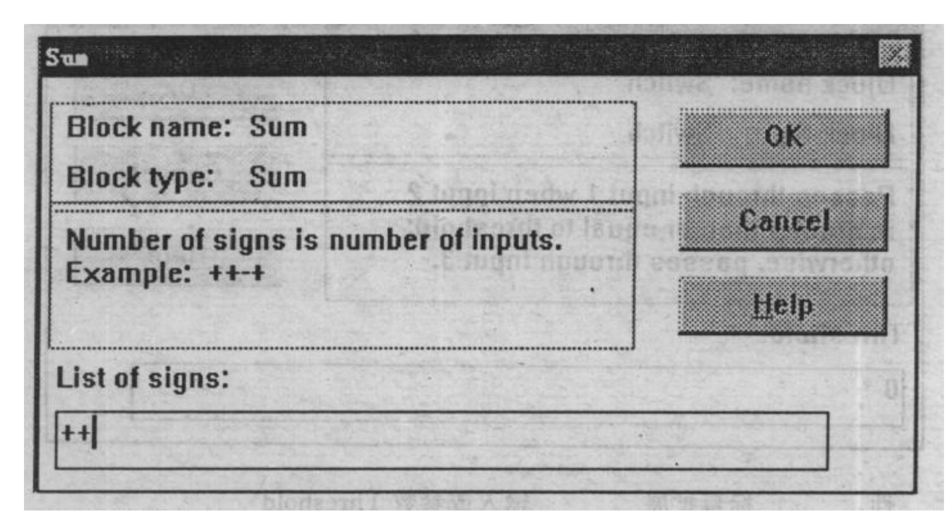
Sum 模块会在其两边显示加、减符号,并和参数 List of signs 中的个数和符号一一对应。如果必要的话,SIMULINK 会调整模块的大小以便显示所有的输入端口。

参　　数　　List of signs

一个常数或一组+、-符号的组合。如果定义了一个常数,SIMULINK 就显示相应个数的输入端口,而且所有的极性都为正;如果定义了一组+、

一符号的组合，那么每一个符号就表示这个端口的极性，符号的个数就等于输入端口的个数。除了加号以外，所有字符包括空格都代表减号。

对 话 框



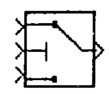
特　　性　　采样时间　　从驱动模块中继承

状　　态　　0

直接输出　　是

58. Switch

图　　标



所在库名　　Nonlinear library

说　　明　　Switch 模块的功能是根据第二个输入来决定输出其它两个输入中的一个。如果第二个输入大于或等于参数 Threshold 中的值，则输出第一个。否则的话，输出第三个输入。

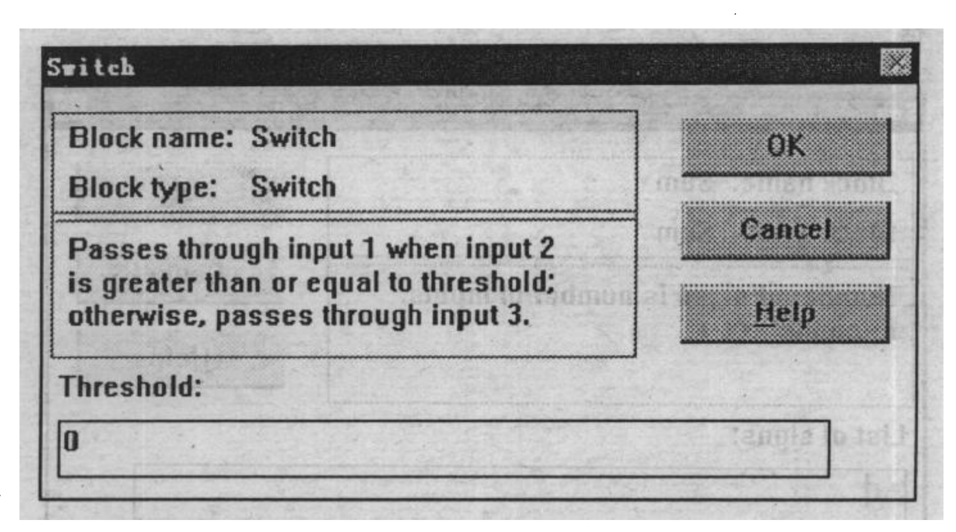
输出信号的维数和输入信号的维数一致。如果所有的输入和参数 Threshold Threshold 是个标量，则输出也是标量。如果所有的输入或者参数 Threshold Threshold 是个向量，则所有的输入向量都必须具有相同的维数。如果是个标量，则要进行标量扩展，形成相应维数的向量。

如果要用 Switch 模块实现一个逻辑开关，用户应该把参数 Threshold 设为 0.5。

参　　数　　Threshold

用来触发开关状态发生改变的控制值。用户可以把这个参数定义为一个标量，也可以是个和输入向量维数相同的向量。如果定义为一个标量，就进行向量扩展。

对 话 框



特　　性	标量扩展	输入或参数 Threshold
	采样时间	从驱动模块中继承
	状　　态	0
	直接输出	是

59. To File

图　　标

untitled.mat

所在库名　Sinks library

说　　明　To File 模块的功能是把输入写到MAT 文件的一个矩阵当中。本模块接收一个输入，它既可以是个标量，也可以是个向量，并为每一步时间对应的数据写成一列：第一行是仿真时间，每列的其它行对应的是输入数据。

To File 模块生成文件的格式和 From File 模块的需要格式一致。因此本模块的输出可以作为 From File 模块的输入。

To File 模块在其图标上显示输出文件的名字。

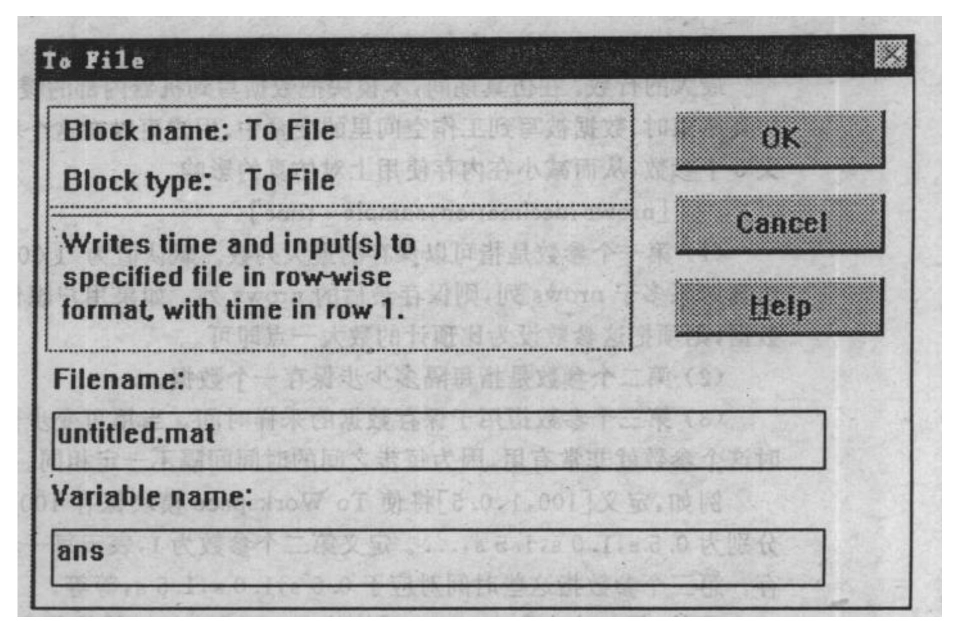
参　　数　Filename

存放矩阵的文件名。如果定义的文件名已经存在了，则其内容就被覆盖。

Variable name

文件中所存放矩阵的名字。

对 话 框



特　　性　　　采样时间　　　从驱动块继承
　　　　　　　状　　态　　　0

60. To Workspace

图　　标

yout

所在库名　　　Sinks library

说　　明　　　To Workspace 模块的功能是把输入写到工作空间里的一个矩阵当中。本模块接收一个输入，它既可以是个标量，也可以是个向量，并默认为每一步标量值或向量中的每一个元素值写成一行。用户可以用参数 Maximum number of rows 来控制写到矩阵中数据点的个数。

如果从 From Workspace 模块来的数据在其它的仿真中需要重新采用，由于要求第一列必须是仿真时间，因此用户可以通过下面两种办法来为数据增加一列仿真时间：

(1) 把 Clock 模块的输出作为 To Workspace 模块输入向量中的第一个元素。

(2) 从命令行或在仿真参数对话框中定义时间为返回值。当仿真结束时，用户可以采用下面的命令来处理：

matrix = [t ; matrix];

本模块在其图标上显示存放其输入数据的矩阵名。

参　　数　　　Variable name

存放数据的矩阵名。如果这个矩阵名在工作面里已经存在了，则本模块覆盖其中的数据。直到仿真结束时，这个存放仿真结果的矩阵才存在。

Maximum number of rows(time steps)

最大的行数。在仿真期间，本模块把数据写到机器内部的缓冲器中。当仿真结束时，数据被写到工作空间里的变量中。用户可以在这个域中最多定义 3 个参数，从而减小在内存使用上对仿真的影响。

[nrows, decimation, sample _ time]

(1) 第一个参数是指可以保存的最大列数。默认值为 '1 000 列。如果仿真结果多于 nrows 列，则保存最后的 nrows 列。如果用户想保存所有的数据，则须把这参数设为比预计的数大一点即可。

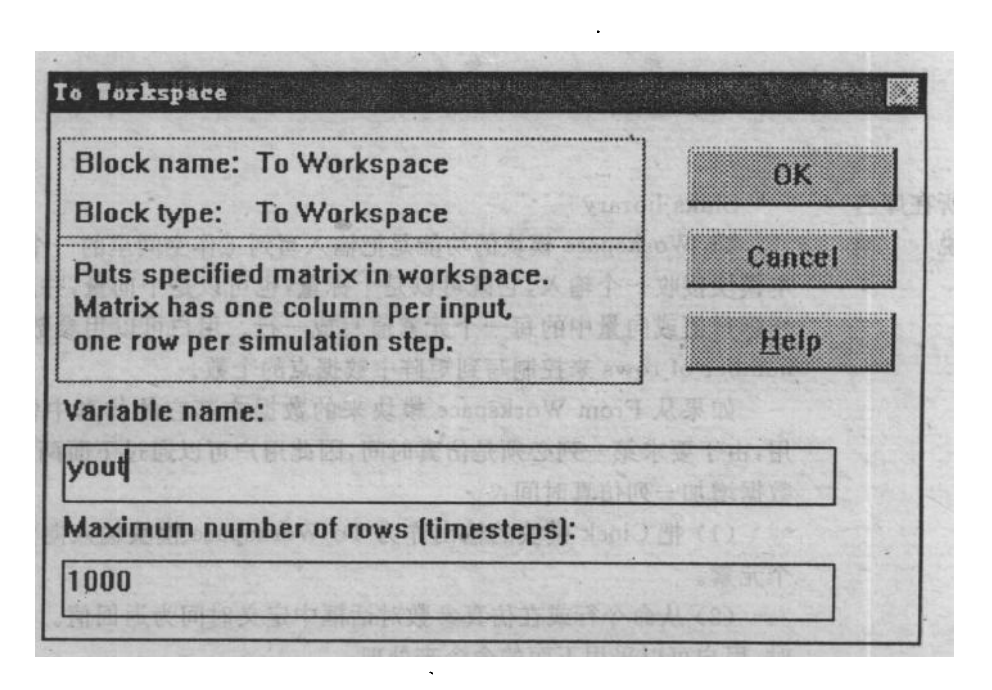
(2) 第二个参数是指每隔多少步保存一个数据。

(3) 第三个参数指用于保存数据的采样时间。当用可变步长进行积分时这个参数就非常有用，因为每步之间的时间间隔不一定相同。

例如，定义[100,1,0.5]将使 To Workspace 模块保存 100 个点，时间分别为 0.5 s,1.0 s,1.5 s,...。定义第二个参数为 1，表示每一步的值都保存。第三个参数指这些时间对应于 0.5 s,1.0 s,1.5 s，等等。

另外，假如定义[100,1,0.5]将使 To Workspace 模块保存 100 个点，时间分别为 2.5 s,5.0 s,7.5 s,...。定义第二个参数为 5，表示每 5 步保存一个。第三个参数指这些时间对应于 2.5 s,5.0 s,7.5 s，等等。

对 话 框



特　　性　　　采样时间　　　离散连续

　　　　　　　状　　态　　　0

61. Transfer Fcn

图　　标

$$\frac{1}{s+1}$$

所在库名　　Linear library

说　　明　　Transfer Fcn 模块的功能是实现一个传递函数，它以下式所示的传递函数形式来表示输入 u 和输出 y 之间的关系：

$$H(s)=\frac{y(s)}{u(s)}=\frac{num(s)}{den(s)}=\frac{num(1)s^{nn-1}+num(2)s^{nn-2}+\cdots+num(nn)}{den(1)s^{nd-1}+den(2)s^{nd-2}+\cdots+den(nd)}$$

其中 nn 和 nd 分别是分子和分母的阶次。行向量 num 和 den 包含有分子和分母两个多项式的系数，并以 s 的递减次幂的形式排列。一个具有 n 个元素的向量定义了一个 n－1 次的多项式。分母的阶次必须大于等于分子的阶次。本模块接受一个标量输入，并产生一个标量输出。

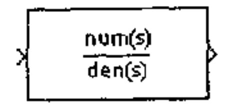
本块模将初始条件被设为零。如果用户必须定义初始条件，请用 State Space 模块。

根据 Transfer Fcn 模块对分子和分母多项式定义的不同，在其图标上显示不同的形式：

(1)如果定义为一个表达式、一个向量、一个括号中的变量，图标就显示定义的参数和相应 s 的幂级数。如果定义了一个用扩号扩起来的变量，首先求变量的值，然后将其值显示出来。例如，如果定义参数 Numerator 为[3,2,1]，参数 Denominator(Den)为[7,5,3,1]，则图标显示为

$$\frac{3s^2+2s+1}{7s^3+5s^2+3s+1}$$

(2)如何定义为一个变量，图标就显示变量名，然后跟“(s)”。例如，如果用户定义参数 Numerator 为 num，参数 Denominator 为 den，则图标显示为



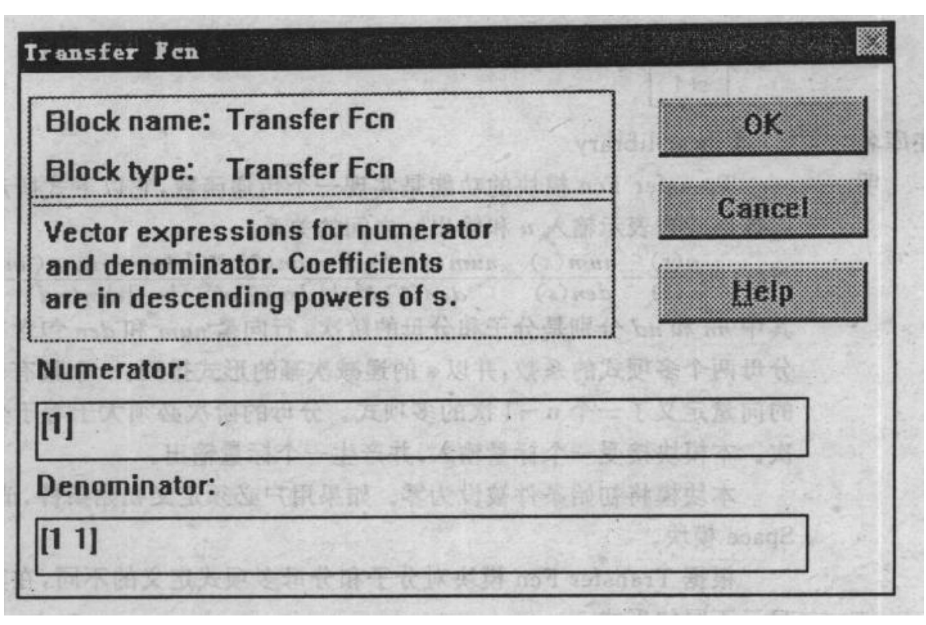
参　　数　　Numerator

分子多项式系数向量。有几行的矩阵表示系统有几个输入。默认值为[1]。

Denominator

分母多项式系数向量。默认值为[1 0.5]。

对 话 框



特　　性	标量扩展	否
	采样时间	连续
	状　　态	其状态个数为分母的阶次
	直接输出	如果分子和分母阶次相同，则输出

62. Transport Delay

图　　标



所在库名　　Nonlinear library

说　　明　　Transport Delay 模块的功能是对输入信号进行规定的延时输出。

在仿真开始的时候，本模块输出参数 Initial Input 中的值，直到仿真时间超过参数 Time Delay 中的值，然后输出延迟的输入值。本模块把前面的输入保存到一个循环缓冲器中，它的大小有参数 Initial Buffer size 来定义。

当需要输出的时候，存储的值与需要的值不对应，本块就在两点之间进行线性插值。因此对精确仿真来说，你应该减小最大步数来增加线性插值的精度。

本模块输入既可以是标量，也可以是个向量。如果是个向量，输出就是个同样维数的向量，SIMULINK 并对标量参数进行标量扩展。

用 linmod 来线性化一个具有 Transport Delay 模块的模型是比较麻烦的。有关更详细的信息参见第七章。

本模块与 Unit Delay 模块的区别在于 Unit Delay 模块不仅延时而且在每一个采样期间保持输出。Unit Delay 模块等价于一个对输入具有零阶采样保持器的 Transport Delay 模块。它们之间的区别可用下面的模型(图

9－3)来演示：

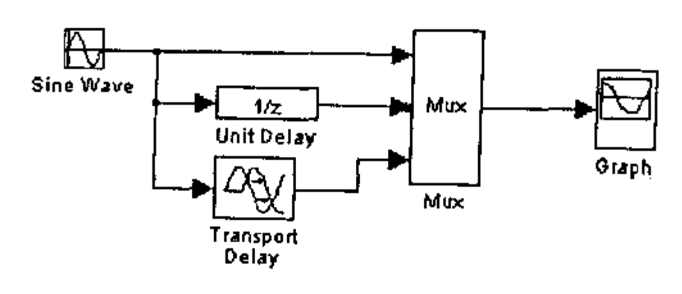


图 9－3

在这个模型中，Unit Delay 模块有一个 1 s 的时间延迟且采样时间为 1 s。Transport Delay 模块有一个 1 s 的时间延迟且初始输入为零。图 9－4 是这个模型的输出曲线：

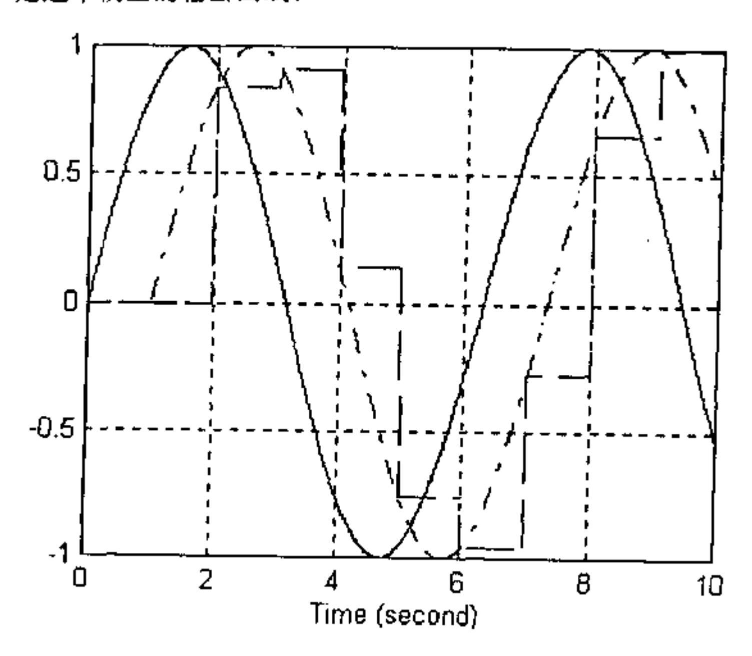


图 9－4

参　数　　Time delay

在把输入信号经过延迟输出以前总的仿真时间。如果定义为一个向量，它的维数必须和输入的维数一致。如果有必要的话，将进行标量扩展。

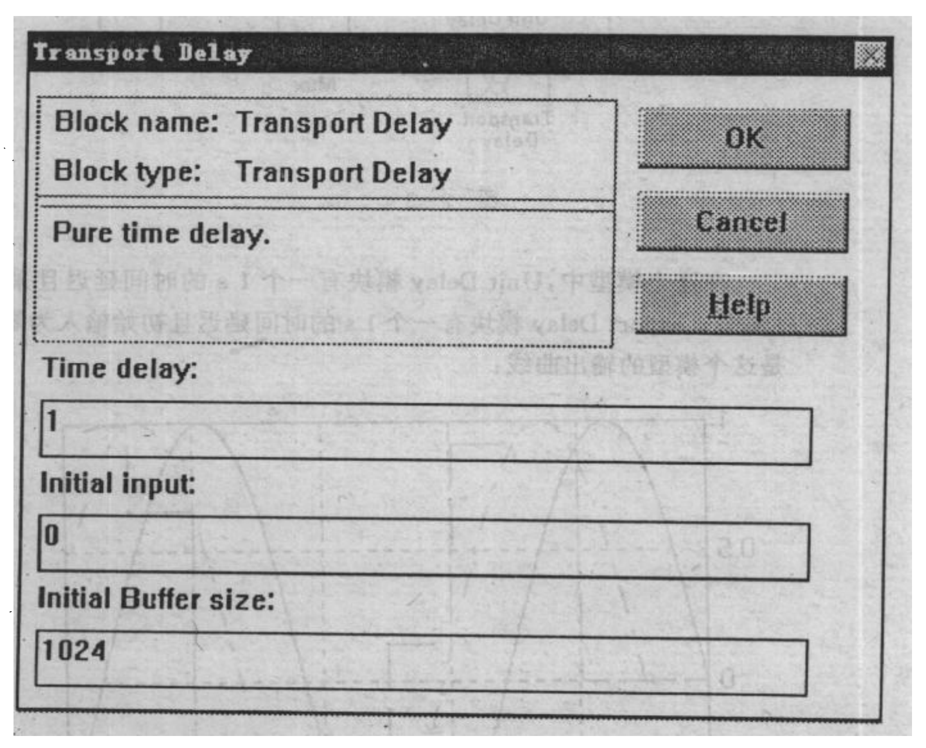
Initial Input

在开始仿真到 Time delay 期间，本模块的输出值。如果定义为一个向量，它的维数必须和输入的维数一致。如果有必要的话，将进行标量扩展。

Initial Buffer size

为存储数据，分配相应个数的初始内存。如果延迟时间比较长，可能需要的内存就比较大，特别是向量输入。如果一个仿真需要更多的缓冲空间，在仿真结束后，就显示有关增大缓冲器大小的建议。

对 话 框



特　　性	标量扩展	参数
	采样时间	连续
	状　　态	0
	直接输出	否

63. 2－D Look－Up Table

图　　标



所在库名　　Nonlinear library

说　　明　　2－D Look－Up Table 模块的功能是对输入分段线性映射成一张由模块中参数定义的表。

用户可以通过参数 X Index 和 Y Index 来定义行和列的指针，用参数 Table 来定义表。本模块通过比较输入和指针 X、Y 来产生一个输出值。

(1)如果输入和指针 X、Y 匹配，则输出指针 X、Y 相交的那个值。

(2)如果输入和指针 X、Y 不匹配，则在相应的指针 X 和/或指针 Y 之间进行插值，并由表中相应元素产生相应的输出。

如果输入中的一个或两个小于第一个或大于最后一个指针，则在最前面两点或最后面两点之间进行外插。

本模块只接收标量输入和输出。有一个有关 2-D Look-Up Table 模块的演示程序，只要在 MATLAB 的命令窗口键入 tabdemo 即可。

参　数　　X Index

指向表的行指针，它是一个行向量或是个列向量。向量值必须单调递增。

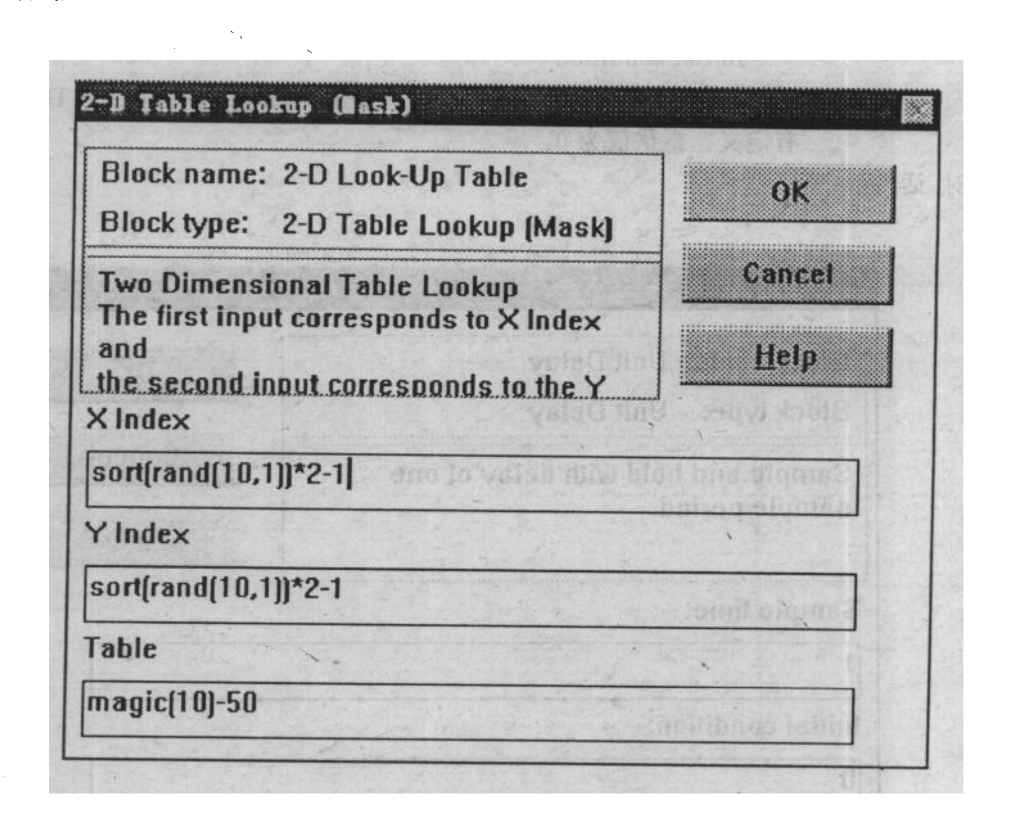
Y Index

指向表的行指针，它是一个行向量或是个列向量。向量值必须单调递增。

Table

由有可能输出的值所组成的一张表。表的大小应与参数 X Index 和 Y Index 对应起来。

对话框



特　性	采样时间	从驱动模块中继承
	状　态	0
	直接输出	是

64. Unit Delay

图　标

1/z

所在库名　　discrete library

说　　明　　Unit Delay 模块的功能是把输入信号进行单位延迟并保持一个采样周期。如果输入信号是个向量，那么向量中的所有元素都延迟相同的时间。本模块等价于离散时间算子 z^{-1}。

如果想要用没有延迟的采样和保持功能，则采用 Zero－Oeder Hold 模块；如果想要大于一个单位的延迟功能，则采用 Discrete Transfer Fcn 模块。

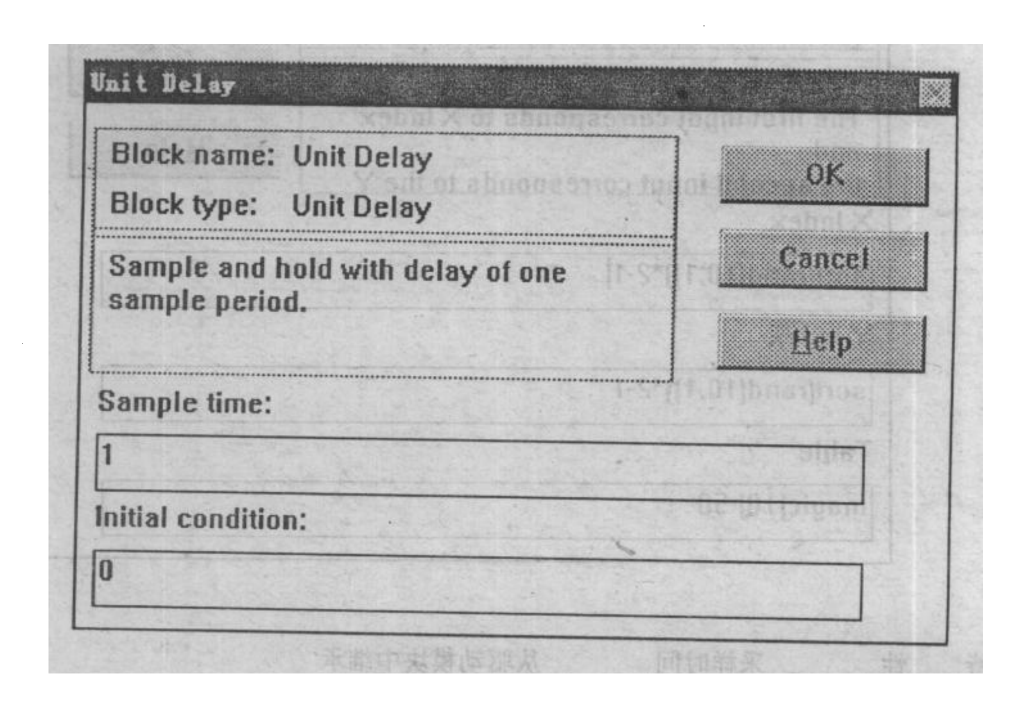
参　　数　　Sample time

采样周期。Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期；第二个元素(如果有的话)是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表采样的延迟，负的偏移表采样的提前。

Initial condition

在第一个采样周期内本模块的输出，在那段时间内 Unit Delay 模块没有定义。默认值为 0。

对 话 框



特　　性　　标量扩展　　参数 Initial condition

采样时间　　离散

状　　态　　从驱动模块中继承

直接输出　　否

65. Variable Transport Delay

图　　标



所在库名　　Nonlinear library

说　　明　　Variable Transport Delay 模块的功能是对输入信号进行时间可变的延迟后再输出。本模块可用来模拟具有管路的系统，其中管路里马达驱动液体的速度是可变的。本模块有两个输入：第一个是通过模块的信号；第二个是延迟时间。在仿真期间，本模块把时间和输入值以一一对应的形式存入一个内部缓冲器中。

在仿真开始的时候，本模块输出参数 Initial Input 中的值，直到仿真时间超过延迟时间。在仿真的每一步，本模块输出的信号是对应于当前时间减去延迟时间的那个信号。

当需要输出的时候，所存储数据的时间与需要的值不对应，本模块就在两点之间进行线性插值。本模块也对开始仿真以前的时间进行外插。

本模块输入既可以是标量，也可以是向量。如果输入的信号是向量，则它们必须具有相同的维数。除非输入信号中没有一个是向量，否则的话，输出一定是向量。SIMULINK 可对任何的标量参数和输入进行标量扩展。

当用户采用一个较小的最大步长或一个固定的步长（这时最大和最小步长大小相同），可以提高结果的精度。如果延迟时间为负数，则用最后两个输入进行线性外插来预测将来的值。

参　　数　　Maximum delay

最大的延迟时间。

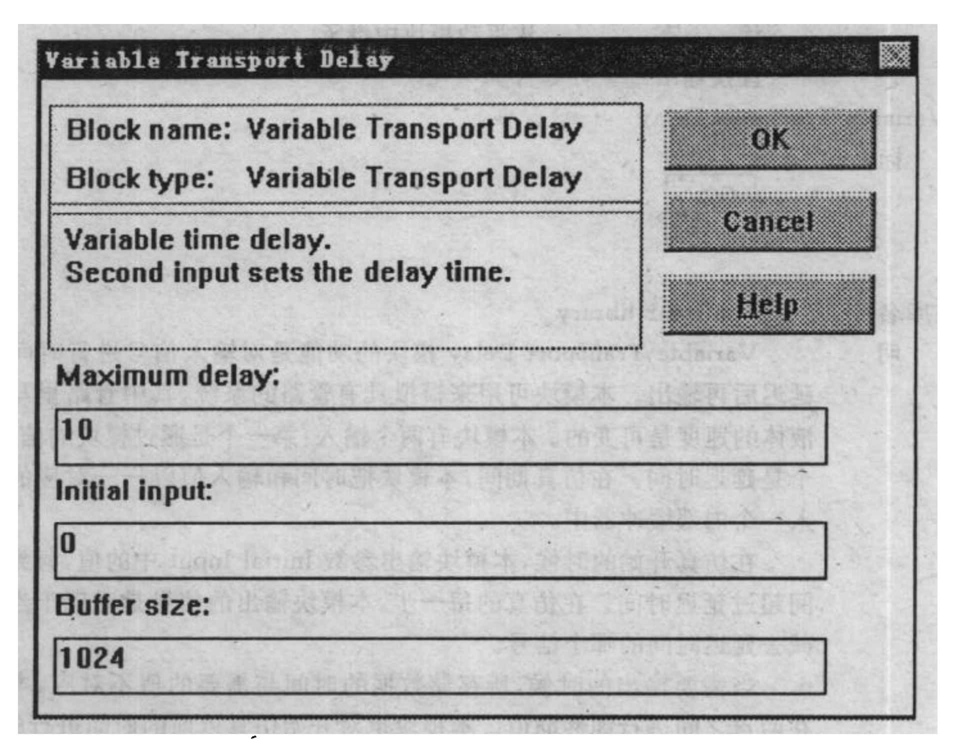
Initial Input

在开始仿真到延迟期间，本模块的输出值。

Initial Buffer size

本模块可以存放的最大个数。如果延迟时间比较长，可能需要的内存就比较大，特别是向量输入。

对话框



特　　性　　采样时间　　连续
　　　　　　状　　态　　0
　　　　　　直接输出　　是

66. XY Graph Scope

图　　标



所在库名　　Sinks library

说　　明　　XY Graph Scope 模块的功能是在 MATLAB 的图形窗口显示输入信号的 X－Y 图。本模块有两个输入，把第一个输入画在 X 方向，把第二个输入画在 Y 方向。本模块对检查极限环和两个状态的数据非常有用。本模块不显示在定义范围以外的数据。

有关 XY Graph Scope 模块的使用有一个演示程序，只要在 MATLAB 的命令窗口键入 lorenzs 即可。

参　　数　　X－min

x 轴的最小值，默认值为－10。

X－max

x 轴的最大值，默认值为 10。

Y－min

y 轴的最小值，默认值为－10。

Y－max

y 轴的最大值，默认值为 10。

对 话 框

XY scope. (Mask)

Block name: XY Graph

Block type: XY scope. (Mask)

XY scope using MATLAB graph window.
First input is used as time base.
Enter plotting ranges.

x-min:

-10

x-max:

10

y-min:

-10

y-max:

10

OK

Cancel

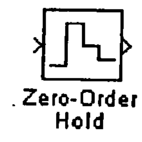
Help

特　　性　　采样时间　　从驱动模块中继承

　　　　　　状　　态　　2 个离散

67. Zero－Order Hold

图　　标



所在库名　　Discrete library

说　　明　　Zero－Order Hold 模块的功能是实现一个以规定的速率进行采样的采样保持器。本模块接收一个输入，并产生一个输出，两者既可以是标量，也可以是向量。

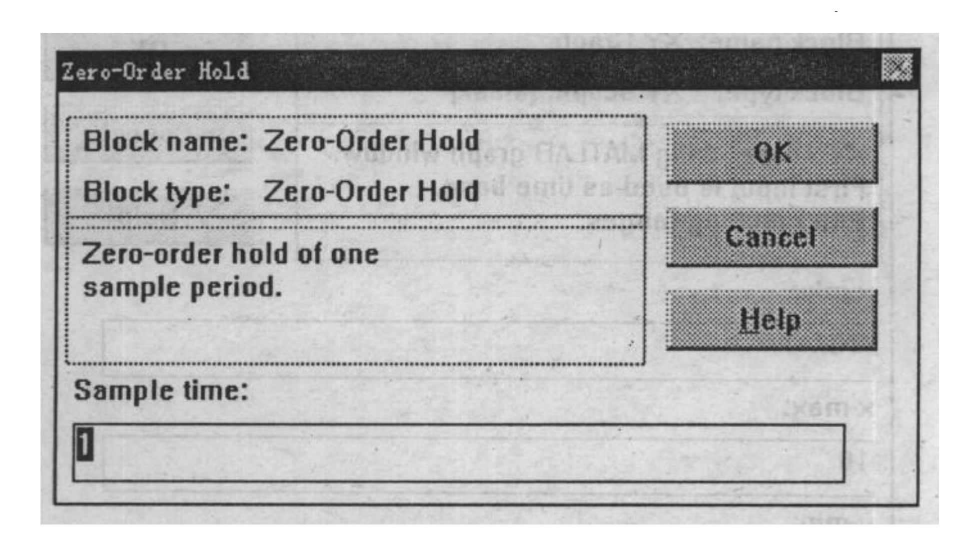
本模块为离散一个或多个信号提供了一种简便的方法。用户可以把它用在不需要其它复杂的离散功能模块，但需要模拟采样这种情况下。例如，

用户可以把它和 Gain 模块联用来模拟一个具有放大器的 A/D 转换器。

参　　数　　Sample time

采样周期。Sample time 可以是标量或是一个具有两个元素的向量。第一个元素是采样周期;第二个元素(如果有的话)是偏移时间。偏移时间允许在规定的采样时间点上偏移一段时间。正的偏移表示采样的延迟,负的偏移表示采样的提前。

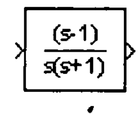
对 话 框



特　　性		
	采样时间	离散
	状　　态	0
	直接输出	是

68. Zero - Pole

图　　标



所在库名　　Linear library

说　　明　　Zero Pole 模块的功能是建立一个具有拉普拉斯算子 s 形式、并具有指定零极点和增益的系统。本模块接受一个标量输入,并产生一个标量输出。

对单输入单输出系统的传递函数可表示为分式形式或零点-极点-增益两种形式,即

$$H(s)=K\frac{Z(s)}{P(s)}=K\frac{(s-Z(1))(s-Z(2))\cdots(s-Z(n))}{(s-P(1))(s-P(2))\cdots(s-P(n))}$$

其中 Z 表示零点向量,P 表示极点向量,K 表示标量增益。

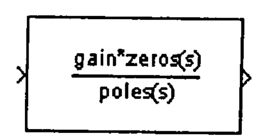
根据 Zero Pole 模块对其参数的定义不同,在其图标上的显示就不同:

(1) 如果定义为一个表达式、一个向量、一个括号中的变量,图标就显示一个具有指定零点、极点和增益的传递函数。如果定义了一个括号括起

来的变量，首先求变量的值，然后显示出来。例如，如果定义参数 Zeros 为[3,2,1]，参数 Poles 定义为变量 poles(poles 为工作空间里定义的矩阵[7,5,3,1])，参数 Gain 为 gain，则图标显示为

$$\frac{gain(s-3)(s-2)(s-1)}{(s-7)(s-5)(s-3)(s-1)}$$

（2）如何定义为一个变量，图标就显示变量名，然后跟“(s)”。例如，如果你定义参数 Zeros 为 zeros，参数 Poles 为 poles，参数 Gain 为 gain，则图标显示为



参　　数　　Zeros

零点向量。默认值为[1]。

Poles

极点向量。默认值为[0;-1]。

Gain

标量增益，可输入一个常数或变量。默认值为[1]。

对 话 框

Zero-Pole

Block name: Zero-Pole

Block type: Zero-Pole

Vector expressions for numerator, denominator, and gain.

OK　Cancel　Help

Zeros:

zeros

Poles:

poles

Gain:

gain

特　　性	采样时间	连续
	状　　态	可变
	直接输出	如果极点数和零点数相同,则输出

第十章 分析命令

本章提供了有关对仿真进行分析的命令的详细参考信息。这些分析命令包括以下类型的命令：

1. 积分方法

(1) linsim。

(2) rk23(Runge - Kutta 23)。

(3) rk45(Runge - Kutta 45)。

(4) admas。

(5) gear。

(6) euler。

2. 线性化分析

(1) linmod：在工作点附近提取一个模型的线性状态空间模型。

(2) dlinmod：在工作点附近提取一个模型的线性、离散状态空间模型。

(3) linmod2：线性化的一种高级形式。

3. 平衡分析

trim：寻找满足输入、输出和状态条件的稳态参数。

10.1 积分方法

10.1.1 语法

[t,x,y]=*method*('*model*',tfinal)

[t,x,y]=*method*('*model*',tfinal,x0)

[t,x,y]=*method*('*model*',tfinal,x0,options)

[t,x,y]=*method*('*model*',tfinal,x0,options,ut)

method('*model*',t)

10.1.2 选项

t,x,y　时间(t)、状态(x)和输出(y)的返回值。如果左边的选项没有定义而模型有输出，那么就以图形的方式显示输出；否则的话，显示状态轨迹。

method　积分方法，包括：linsim rk23 rk45 adams gear euler

model　模型名，用单引号括起来。

tfinal　仿真开始和结束时间。如果把它定义为一个标量，那么 tfinal 就是结束时间，而开始时间为 0；如果把它定义为一个具有两个元素的向量[tstart tfinal]，就分别定义开始时间和结束时间。

x0　初始条件向量。如果定义了这个值，那么就忽视在块中定义的初始条件，除非 x0 是一个空矩阵。

options　一个具有 6 个元素的可选的仿真参数向量。如果这个值没有定义或是空向量或

全为 0,那么就用其默认值。

options(1) 容许误差,默认值为 1e-3。

options(2) 最小积分步长,默认值为 tfinal/2 000。

options(3) 最大积分布长,默认值为 tfinal/50

options(4) 该值为 1,选择 Adams/Gear。根据模型刚性程度,选择适当的积分方法(要么是 Adams,要么是 Gear)。

options(5) 如果该值不为 0,那么当达到最小步长时,就显示一个警告信息,指出容许误差不能得到满足。当系统中有不连续性时,这种情况就有可能发生。默认值是不要警告信息。

options(6) 画图控制参数(当为 1 时画图,当为 2 时不画图)。默认值是当左边没有选项时就进行画图。

ut 外部输入,ut 已可以是字符串,也可以是一张由值组成的表。如果定义为一个字符串,MATLAB 就计算其值,然后把所得的结果作为系统的外部输入。这个字符串可以是包含有工作空间里变量的表达式。例如,定义'sin(t)',那么在每一步,输入就是 sin(t)。如果定义为一个矩阵,那么第一列必须是一个按照升序形式排列的时间向量,后面的 n 列必须是 n 个输入对应的输入值。SIMULINK 在需要的时候就在时间、输入之间进行线性化插值。例如,下面这些命令就可以得到外部输入值:

```
t=0:0.1:10;
u=[cos(t),sin(t),tam(t)];
ut=[t,u];
rk45(model,t,[ ],[0,min,max],ut];
```

10.1.3 说明

有了左边三个选项,积分函数就返回仿真时间 t、状态轨迹 x 和输出轨迹 y。如果没有左边三个选项,那么命令就对 model 中描述的常微分方程在 0 到 tfinal s 之间对模型进行积分。如果模型有输出,就以图形的方式显示,否则的话就显示状态轨迹。

步长控制:

所有模型的积分步长都是采用不同算法的变步长方法。仿真时,需不断调整步长的大小来满足容许误差的要求。步长实际上指定了用来产生输出的两点之间的最小距离。

用于产生输出和状态轨迹的步长总是满足 options(2)≤步长≤optins(3)。实际上,除了 gear 和 adams 以外,所有的算法都可变成固定步长的方法,方法是把最小步长,即 optins(2)设得和最大积分步长,即 optinon(3)一样大。linsim 和 euler 是每步产生一个输出的单步方法。rk23 和 rk45 是在产生输出的两点之间需要计算的龙格库塔方法。

(1) rk23 是一个三阶的方法。而对阶跃型控制,它采用二阶方法。要产生一个输出,rk23 需要分三步运算,它的最小步长为 options(2)/2。

(2) rk45 是一个 5 阶的方法。而对阶跃型控制,它采用 4 阶方法。要产生一个输出,它需要 6 个不均匀分布的步长来计算输出,它的最小步长为 options(2)/13 adams 和 gear 这两种方法称之为预测—校正法。它采用可变的步数来产生一个输出。

10.1.4 举例

下面这条命令描述的是 Van-der-Pol 模型。向量 sizes 显示的是模型特性(其中的注释

指出每个向量元素的含义)，向量 x0 显示的是初始条件，向量 str 指出的是状态(和初始条件)的顺序。

$$[sizes,x0,str] = vdp([\ ],[\ ],[\ ],0)$$

$$sizes = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$x0 = \begin{bmatrix} 0.25 \\ 0.25 \end{bmatrix}$$

$$str = \begin{bmatrix} /vdp/int\ x2 \\ /vdp\ int\ x1 \end{bmatrix}$$

要想证实上面这些结果，在命令窗口键入 vdp 即可。模型中的块名是隐含的，当然也可使它们显示出来，方法是：首先选中 Integrator 块，然后在 Style 菜单中的 Title 选项中选择 Displayed即可。

要想在命令窗口进行仿真、打印状态和时间的关系曲线并使用块中的初始条件，可以键入：

```
rk45('vdp',10,[],[1e-3 .01 .1])
```

第三个选项定义为一个空矩阵，这样就使得积分器采用块中定义的初始化参数。

要想忽视模块中的初始化参数，可以采用下面的命令：

```
rk45('vdp',10,[2.5 2.5],[1e-3 .01 .1])
```

这条命令使 Integrator 模块的初始条件为 2.5。

10.2 线性化分析

10.2.1 语法

[A,B,C,D] = *linfun*('*model*')

[A,B,C,D] = *linfun*('*model*',x,u)

[A,B,C,D] = *linfun*('*model*',x,u,pert)

[A,B,C,D] = *linfun*('*model*',x,u,pert,xpert,upert)

10.2.2 选项

linfun 线性化函数，包括：linmod，dlinmod linmod2。

model 需要线性化的模型名称。

x 和 u 状态和输入向量。如果定义这些量，那么这些量就设置了工作点，在这个点上提取线性模型。

pert 可选标量，用来设置 x 和 u 的干扰。如果这个值没有定义，就采样默认值 1e-5。

xpert 和 upert 可选向量，用来为每一个状态和输入设置扰动。如果定义了这个向量，那么就忽视上面的标量 pert，在这种情况下：

状态 x 的第 i 个元素受到扰动后就变成 x(i)+xpert(i)；

输入 u 的第 j 个元素受到扰动后就变成 u(j)+upert(j)。

10.2.3 说明

linmod 得到的是用常微分方程描述的 SIMULINK 模型的线性模型。返回的模型是以状态空间 A、B、C、D 形式来表示其输入和输出之间的关系，这样的线索性模型可用下面的式子来表示：

[A,B,C,D]=linmod('model')可以得到在状态变量 x=0 和输入 u=0 这个工作点附近的线性化模型。

linmod 是在工作点附近对状态施加扰动后来确定状态导数和输出的变化速率(即雅可比矩阵)，并把得到的结果用来计算状态空间矩阵。每一个状态 x(i)收到扰动后变成：

$$x(i)+\Delta(i)$$

其中

$$\Delta(i)=\delta(1+|x(i)|)$$

同样，第 j 输入受到扰动后，变成

$$u(j)+\Delta(j)$$

其中

$$\Delta(j)=\delta(1+|u(j)|)$$

10.2.4 离散系统的线性化

dlinmod 能够以任意给定的采样时间对离散系统、多速率系统以及连续和离散这类混合系统进行线性化。除了第二个选项需要插入采样时间来对系统线性化以外，dlinmod 的调用格式和 linmod 是相同的。例如：

[Ad,Bd,Cd,Dd]=dlinmod('model',Ts,x,u)

上面这条命令可以产生一个以状态向量 x 和输入向量 u 为工作点、采样时间为 Ts 的离散状态空间模型。要想得到一个离散系统的近似连续模型，只要把 Ts 设为 0 即可。

如果一个模型是由线性模块、多速率模块、离散模块和连续模块组成的，那么在满足下列条件的情况下，dlinmod 产生一个采样时间为 Ts、且具有相同频率和时间响应的线性模型：

(1) Ts 是系统中所有采样时间的整数倍。

(2) Ts 不小于系统中最慢的采样时间。

(3) 系统是稳定的。

不过，在上面这些条件不满足的情况下，也有可能得到有效的线性模型。要看一个系统是否稳定，实际上只要计算线性化后所得矩阵 Ad 的特征值就可以了。因此，如果 Ts>0 而且特征值在单位圆内，即

all(abs(eig(Ad)))<1

那么系统是稳定的。同样，如果 Ts=0，而且所有的特征值在左半平面，那么系统也是稳定的，即

all(abs(eig(Ad)))<0

当系统不稳定且采样时间不是其它采样时间整数倍的时候，dlinmod 就可能产生复矩阵 Ad 和 Bd。然而在这种情况下，仍然可以通过矩阵 Ad 的特征值来验证系统的稳定性。

dlinmod 可以用来把一个系统的采样时间变成其它值，或者把一个线性离散系统变成一

个连续系统，或者反过来，把一个连续系统变成一个离散系统。

如果需要求一个连续系统的频率响应，可用 MATLAB 的 bode 命令。

线性化的一种高级形式：

程序 linmod2 提供了一种线性化的高级形式，这个程序和 linmod 相比，运行需要更长的时间，但产生的结果要比 linmod 精确。

linmod2 的调用格式和 linmod 调用格式相似，但功能不同。例如，linmod2('model',x,u)和 linmod 一样产生一个线性模型。然后，状态空间矩阵的每一个元素的干扰是逐个设置的，以便使与舍入误差和截断误差有关的误差达到最小。

linmod2 设法平衡舍入误差(由于小扰动引起的，并与计算精度有关的误差)与截断误差(由于大扰动引起的，使分段线性近似无效而产生的误差)，并在两者之间进行折中。

对于调用命令[A,B,C,D]=linmod2('model',x,u,pert)，变量 pert 表示可以施加的最小扰动，默认值为 1e－8。linmod2 的优点是能够探测到不连续性，并产生一个警告信息，例外：

warning:discontinuity detected ad A(2,3)

当出现上述这样的警告信息时，可在其它的工作点附近进行试验来得到线性化模型。

对于调用命令

[A,B,C,D]=linmod2('model',x,u,pert,Apert,Bpert,Cpert,Dpert)

其中变量 Apert、Bpert、Cpert、Dpert 是用来为每一个状态和每个输入的组合而设置的干扰矩阵。因此，xpert 的第 ij 个元素是与矩阵 A 的第 ij 个元素有关的干扰，默认的干扰可用下面的命令来获得：

[A,B,C,D,Apert,Bpert,Cpert,Dpert]=linmod2('model',x,u)

10.2.5 注释

一般来说，系统时间的默认值为 0。因此，对于依赖于时间的系统，变量 pert 可以定义为一个具有两个元素的向量，其中这个向量的第二个元素是用来到设置时间 t 的值，从而得到这个时间值上的线性模型。

当一个需要进行线性化的模型本身就是一个线性模型时，那么就不存在截断误差问题.因此，扰动就可以定义为一个任意希望的值。一般来说，总是希望把这个值定义得大一些，这样将有助于减小舍入误差。另外，对于本身就是线性的模型，所选用的工作点将不影响所得到的线性模型。

一个非线性模型通过线性化后所得到的线性模型，其状态阶数将保持不变。对于 SIMULINK 系统，与每个状态有关的、包含有块名的一个字符串变量可以用下面的命令来获得：

[sizes,x0,xstring]=model

其中 xstring 是一个字符串变量，它的第 i 行是与第 i 个状态有关的块名。输入和输出按照框图中的顺序进行编号。对于单输入多输出系统，可采用 ss2tf 来把它转换成传递函数形式，或用 ss2zp 把它转换成零极点形式。

对于包含有 Derivative 块和 Transport Delay 块的系统，进行线性化是比较麻烦的。有关其详细的描述，请参阅第七章 7.3"线性化"这一节。

10.3 平衡分析

10.3.1 格式

[x,u,y,dx] = trim('*model*')

[x,u,y,dx] = trim('*model*',x0,u0,y0)

[x,u,y,dx] = trim('*model*',x0,u0,y0,ix,iu,iy)

[x,u,y,dx] = trim('*model*',x0,u0,y0,ix,iu,iy,dx0,idx)

[x,u,y,dx] = trim('*model*',x0,u0,y0,ix,iu,iy,dx0,idx,options)

[x,u,y,dx] = trim('*model*',x0,u0,y0,ix,iu,iy,dx0,idx,options,t)

[x,u,y,dx,options] = trim('*model*',...)

[x,u,y,dx] = trim2('*model*',...)

[x,u,y,dx] = trim3('*model*',...)

[x,u,y,dx] = trim4('*model*',...)

10.3.2 说明

trim 的功能是试图找到输入 u 和状态 x 的值来使状态的导数为 0。这样的工作点被称作为平衡点,即系统在稳态时的工作点。

既然这样的平衡点通常不是唯一的,因此有必要对状态 x、输入 u 和输出 y 的值进行程固定。

命令[x,u,y]=trim('model') 的功能是设法找到一个平衡点,使得[x;u;y]的绝对值达最小。

命令[x,u,y]=trim('model',x0,u0,y0)为状态 x、输入 u 和输出 y 定义了各自初始的猜测值。在这种情况下,trim 就使[x-x0;u-u0;y-y0]的最大绝对值达到最小。

x、u、y 的每一个元素可用下面的调用格式进行固定:

trim('model',x0,u0,y0,ix,iu,iy)

整数向量 ix、iu 和 iy 用来指出对 x0、u0 和 y0 的哪一个元素进行固定。既然无法保证平衡点一定存在,因此问题就转化为寻找一个稳态值使得:

abs([x(ix)]-x0(ix);u(iu)-u0(iu);y(iy)-y0(iy)])

最大绝对值达到最小。

trim 在限制状态导数为零的情况下,用一个有约束的优化算法来求解一个由 x、u 和 y 的希望值所组成的最大最小问题。对于这样一个问题,有可能没有可行解。在这种情况下,trim 就在状态导数偏离 0 这种最坏条件下,使上面的最大绝对值达到最小。

要使导数固定为一个非 0 值,可以采用

[x,u,y,dx]=trim('model',x0,u0,y0. ix,iu,iy,dx0,idx)

其中 dx0 表示希望的偏离值,idx 用来表示对 dx 中的哪一个元素进行固定。

10.3.3 举例

考虑如下的一个线性状态空间模型:

$$\dot{x}=Ax+Bu$$

$$y=Cx+Du$$

假定这个模型的名字为 'model',且矩阵 A、B、C、D 如下式所示:

```
A=[  -0.09   -0.011    0  ];
B=[  0     -7     0     -2  ];
C=[  0      2     1     -5  ];
D=[  -3     0     1     0  ];
```

例 1 要寻找一个系统的平衡点，可用下面的命令：

```
[ x,u,y,dx,options ] = trim('model')
```

$$x=\begin{bmatrix}0\\0\end{bmatrix},\quad u=0$$

$$y=\begin{bmatrix}0\\0\end{bmatrix},\quad dx=\begin{bmatrix}0\\0\end{bmatrix}$$

需要的迭代次数为：

```
options(10)
ans =7
```

例 2 要寻找系统在 x=[1;1]和 u=[1;1]附近的一个平衡点，可用下面的命令：

```
x0=[1;1];
u0=[1;1];
[x,u,y,dx,options] = trim('model',x0,u0);
```

$$x=\begin{bmatrix}1.0e-11 *\\-0.116\ 7\\-0.116\ 7\end{bmatrix},\quad u=\begin{bmatrix}0.333\ 3\\0.000\ 0\end{bmatrix},$$

$$y=\begin{bmatrix}-1.000\ 0\\0.333\ 3\end{bmatrix},\quad dx=\begin{bmatrix}1.0e-11 *\\0.421\ 4\\0.000\ 3\end{bmatrix}$$

需要的迭代次数为：

```
options(10)
ans =25
```

例 3 要寻找一个系统输出固定为 1 的平衡点，可用下面的命令：

```
y = [ 1;1 ];
iy = [ 1;2 ];
[ x,u,y,dx ] = trim('model',[],[],y,[],[],iy)
```

$$x=\begin{bmatrix}0.000\ 9\\-0.307\ 5\end{bmatrix},\quad u=\begin{bmatrix}-0.538\ 3\\0.000\ 4\end{bmatrix},$$

$$y=\begin{bmatrix}1.000\ 0\\1.000\ 0\end{bmatrix},\quad dx=\begin{bmatrix}1.0e-16 *\\-0.017\ 3\\0.239\ 6\end{bmatrix}$$

例 4 要寻找一个系统输出固定为 1、状态导数分别为 0 和 1 为的平衡点，可用下面的命令：

```
y = [ 1;1 ];
iy = [ 1;2 ];
dx = [ 0;1 ];
idx = [ 1;2 ];
[ x,u,y,dx,options ]=trim('model',[],[],y,[],[],iy,dx,idx)
```

$$x=\begin{bmatrix}0.9752\\-0.0827\end{bmatrix},\quad u=\begin{bmatrix}-0.3884\\-0.0124\end{bmatrix}$$

$$y=\begin{bmatrix}1.0000\\1.0000\end{bmatrix},\quad dx=\begin{bmatrix}0.0000\\1.0000\end{bmatrix}$$

需要的迭代次数为：

```
options(10)
ans =13
```

参考文献

1 Math Works . MATLAB Version 4.2 User's Guide. 1995

2 张培强编著. MATLAB 语言——演草纸式的科学计算语言. 合肥:中国科技大学出版社,1995

3 薛定宇著. 控制系统计算机辅助设计——MATLAB 语言及应用. 北京:清华大学出版社,1996